Solaris X Window System Reference Manual

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Preface

OVERVIEW

A man page is provided for both the naive user, and sophisticated user who is familiar with the X Window system and is in need of on-line information. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

The following contains a brief description of each section in the man pages and the information it references:

- Section 1 describes, in alphabetical order, commands available with the X Window system.
- Section 3 describes functions found in various libraries.
- Section 4 outlines the formats of various files.
- Section 6 describes various games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals, and device drivers.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the intro pages for more information and detail about each section, and **man**(1) for more information about man pages in general.

NAME

This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS

This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full pathname is shown. Literal characters (commands and options) are in **bold** font and variables (arguments, parameters and substitution characters) are in *italic* font. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

[ ]	The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument *must* be specified.
. . .	Ellipses. Several values may be provided for the previous argument, or the previous argument can be specified multiple times, for example, '*filename* . . .'.
|	Separator. Only one of the arguments separated by this character can be specified at time.
{ }	Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

AVAILABILITY

This section briefly states any limitations on the availabilty of the command. These limitations could be hardware or software specific.

A specification of a class of hardware platform, such as **x86** or **SPARC**, denotes that the command or interface is applicable for the hardware platform specified.

In Section 1 and Section 1M, **AVAILABILITY** indicates which package contains the command being described on the manual page. In order to use the command, the specified package must have been installed with the operating system. If the package was not installed, see **pkgadd**(1) for information on how to upgrade.

DESCRIPTION

This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, functions and such, are described under USAGE.

OPTIONS

This lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.

RETURN VALUES

If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared as **void** do not return values, so they are not discussed in RETURN VALUES.

ERRORS

On failure, most functions place an error code in the global variable **errno** indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.

USAGE

This section is provided as a *guidance* on use. This section lists special rules, features and commands that require in-depth explanations. The subsections listed below are used to explain built-in functionality:

Commands
Modifiers
Variables
Expressions
Input Grammar

EXAMPLES

This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command line entry and machine response is shown. Whenever an example is given, the prompt is shown as

example%

or if the user must be super-user,

example#

Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS and USAGE sections.

ENVIRONMENT

This section lists any environment variables that the command or function affects, followed by a brief description of the effect.

FILES

This section lists all filenames referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.

SEE ALSO

This section lists references to other man pages, in-house documentation and outside publications.

DIAGNOSTICS

This section lists diagnostic messages with a brief explanation of the condition causing the error. Messages appear in **bold** font with the exception of variables, which are in *italic* font.

WARNINGS

This section lists warnings about special conditions which could seriously affect your working conditions — this is not a list of diagnostics.

NOTES

This section lists additional information that does not belong anywhere else on the page. It takes the form of an *aside* to the user, covering points of special interest. Critical information is never covered here.

BUGS

This section describes known bugs and wherever possible suggests workarounds.

NAME

Intro – introduction to the Solaris X Window System Reference Manual

OVERVIEW

The Solaris X Window System is the default windowing and display system included with the Solaris computing environment. It consists of Sun's implementation of the MIT X server and a collection of standard X client applications such as **xlogo** and **xterm.**

The *Solaris X Window System Reference Manual* describes the commands and utility programs included with version 3.5 of the Solaris X Window System. This manual and the *OpenWindows Desktop Reference Manual* cover all man pages that make up the **SUNWaman** software package, other than those in Section 3. This package is usually installed as part of the "Developer" software set during initial installation of Solaris. The default installation directory for these man pages is **/usr/openwin/man.** To access these man pages, add **/usr/openwin/man** to the **$MANPATH** environment variable.

Many of these man pages are from sources outside of Sun. The Display PostScript man pages are from Adobe Systems and most of the X client man pages are from MIT. The original text in these man pages has been left unaltered except necessary corrections for path differences. The Solaris X server uses a somewhat different file hierarchy than the standard MIT X server which stores files in the **/usr/lib/X11** directory. Under Solaris these files have been moved to **/usr/openwin.** Thus the pathname for **/usr/lib/X11/Xinitrc** has been changed to **/usr/openwin/lib/Xinitrc**.

Name	**Command Description**
accessx(1)	a graphical interface to the accessx keyboard enhancements.
appres(1)	list application resource database
atobm(1)	See **bitmap**(1)
bdftopcf(1)	convert font from Bitmap Distribution Format to Portable Compiled Format
bdftosnf(1)	BDF to SNF font compiler for X11
bitmap(1)	bitmap editor and converter utilities for the X Window System
bldrgb(1)	simple interface to the rgb(1) utility
bmtoa(1)	See **bitmap**(1)
cmap_alloc(1)	allocate default colormaps for non-default visuals
cmap_compact(1)	colormap configuration utility to reduce colormap flashing.
constype(1)	print type of Sun console
cps(1)	construct C to language interface
dps(7)	Display PostScript imaging for the X Window System
dpsexec(6)	Display PostScript Executive
editres(1)	a dynamic resource editor for X Toolkit applications
fontadd(6)	See **font_utils**(6)

fontadmin(6)	font administrator
fontls(6)	See **font_utils**(6)
fontrm(6)	See **font_utils**(6)
font_utils(6)	remove installed fonts from a directory
fpadd(6)	See **fp_utils**(6)
fpls(6)	See **fp_utils**(6)
fprm(6)	See **fp_utils**(6)
fp_utils(6)	remove a font path element
fs(1)	X font server
fsadmin(1)	font server administration utility
fsinfo(1)	font server information utility
fslsfonts(1)	server font list displayer for X font server
fstobdf(1)	BDF font generator
ico(6)	animate an icosahedron
ico2(6)	animate an icosahedron or other polyhedron
imake(1)	C preprocessor interface to the make utility
kbd_mode(1)	change the keyboard translation mode
keytable.map(4)	maps keyboard type and layout to keytable
listres(1)	list resources in widgets
makebdf(1)	create bitmap files from scalable F3 or X11 / NeWS font files
makedepend(1)	create dependencies in makefiles
makepsres(1)	Build PostScript resource database file.
maze(6)	an automated maze program
mkdirhier(1)	makes a directory hierarchy
mkfontdir(1)	create fonts.dir file from directory of font files
muncher(6)	draw interesting patterns in an X window
oclock(1)	display time of day
plaid(6)	paint some plaid-like patterns in an X window
pswrap(1)	creates C procedures from segments of PostScript language code
puzzle(6)	15-puzzle game for X
ras2ps(1)	converts a Sun RasterFile to a PostScript file
rasterfile(4)	Sun’s file format for raster images
redxblue(1)	swap red and blue for a 24 or 32 bit rasterfile
resize(1)	utility to set TERM and terminal settings to current window size
rgb(1)	build the color name database

sessreg(1)	manage utmp / wtmp entries for non-init clients
showfont(1)	font dumper for X font server
showrgb(1)	display the color name database
showsnf(1)	display contents of an SNF file
texteroids(6)	test your mousing skills on spinning text
twm(1)	Tab Window Manager for the X Window System
viewres(1)	graphical class browser for Xt
winsysck(1)	check which window system protocols are available
worm(6)	draw wiggly worms
X(7)	See **X11**(7)
X11(7)	a portable, network-transparent window system
xauth(1)	X authority file utility
xbiff(1)	mailbox flag for X
xcalc(1)	scientific calculator for X
xclipboard(1)	X clipboard client
xclock(1)	analog / digital clock for X
xcmsdb(1)	Xlib Screen Color Characterization Data utility
xcolor(1)	displays 256 colors in an X window.
xconsole(1)	monitor system console messages
xcutsel(1)	interchange between cut buffer and selection
xditview(1)	display ditroff DVI files
xdm(1)	X Display Manager with support for XDMCP
xdpr(1)	dump an X window directly to a printer
xdpyinfo(1)	display information utility for X
xedit(1)	simple text editor for X
xepsf(6)	display an Encapsulated PostScript file
xev(6)	print contents of X events
xeyes(6)	Eyes follow your pointer
xfd(1)	display all the characters in an X font
xfontsel(1)	point & click interface for selecting X11 font names
xgc(6)	X graphics demo
xhost(1)	server access control program for X
xinit(1)	X Window System initializer
xkill(1)	kill a client by its X resource
xload(1)	system load average display for X

xlock(1)	locks the local X display until a password is entered
xlogo(1)	X Window System logo
xlsatoms(1)	list interned atoms defined on server
xlsclients(1)	list client applications running on a display
xlsfonts(1)	server font list displayer for X
xlswins(1)	server window list displayer for X
xmac(6)	display Apple MacPaint image files under X windows
xmag(1)	magnify parts of the screen
xmag_multivis(6)	magnify parts of the screen
xmakemap(1)	make a keyboard mapping to be used as input to xmodmap
xman(1)	manual page display program for the X Window System
xmh(1)	send and read mail with an X interface to MH
xmkmf(1)	simple interface to the imake utility for generating X11 Makefiles
xmodmap(1)	utility for modifying keymaps in X
xpr(1)	print an X window dump
xprop(1)	property displayer for X
xrdb(1)	X server resource database utility
XReadScreen(3)	returns the displayed colors in a rectangle of the screen
xrefresh(1)	refresh all or part of an X screen
xscope(6)	X Window Protocol Viewer
Xserver(1)	X Window System server
xset(1)	user preference utility for X
xsetroot(1)	root window parameter setting utility for X
xsol(6)	play solitaire
XSolarisGetVisualGamma(3)	obtains gamma information for a visual
XSolarisOvlCopyAreaAndPaintType(3)	copies the given area and paint type data from one pair of drawables to another
XSolarisOvlCopyPaintType(3)	renders opaque and transparent paint into the destination drawable based on the paint type attributes of the pixels in the source drawable
XSolarisOvlCreateWindow(3)	creates an overlay window
XSolarisOvlGetPaintType(3)	gets the current paint type set in the GC

XSolarisOvlIsOverlayWindow(3)	indicates whether a given window is an overlay window
XSolarisOvlSelectPair(3)	selects an optimal overlay / underlay visual pair that best meets the criteria
XSolarisOvlSelectPartner(3)	returns the overlay / underlay visual that best meets the criteria
XSolarisOvlSetPaintType(3)	specifies the type of paint rendered by subsequent Xlib drawing with the given GC
XSolarisOvlSetWindowTransparent(3)	sets the background state of an overlay window to be transparent
xstdcmap(1)	X standard colormap utility
Xsun(1)	Solaris server for X Version 11
xterm(1)	terminal emulator for X
xwd(1)	dump an image of an X window
xwininfo(1)	window information utility for X
xwud(1)	image displayer for X

NAME

accessx – a graphical interface to the accessx keyboard enhancements.

SYNOPSIS

accessx [**-o**] [**-i**] [**-a**]

AVAILABILITY

The keyboard enhancements and corresponding graphical interface are available on any UNIX workstation running the OpenWindows version 3.4 or later.

DESCRIPTION

accessx is a graphical user interface to enhancements within the OpenWindows 3.4 server. These enhancements provide the keyboard with additional capabilities which are primarily designed to aid users with disabilities who are unable to use a keyboard or mouse in the usual fashion.

OPTIONS

-o Each user may load and save personal default settings for the various keyboard enhancement features. By default, the **accessx** utility will read in the user's default settings and configure the system to those settings. However, in certain instances the user might wish to use the systems current settings instead of his / her own default settings. Specifying the -o option will instruct the **accessx** utility to ignore a user's default settings and use the systems current settings instead.

-i When this option is specified, the **accessx** user interface initially appears in an iconified state.

-a Users may pop up dialog boxes indicating the state of Sticky Keys and Mouse Keys via the main menu of the **accessx** utility's graphical interface. Some users, however, want these status dialog windows to appear and disappear automatically whenever they turn on and off Sticky Keys or Mouse Keys. Specifying the -a option instructs the **accessx** utility to automatically pop up and down the Sticky Keys status window whenever Sticky Keys is turned on or off, respectively, and to automatically pop up and down the Mouse Keys status window whenever Mouse Keys is turned on or off, respectively.

ENVIRONMENT

The HOME environment variable is used to identify the directory in which to load and save a user's default settings. The settings are stored in X resource file format in the file $HOME / .AccessX. The user may modify or add resources to this file directly, however all comments will be deleted when the user performs a save settings action. Note that the UIDPATH environment variable should be set to **/usr/openwin/lib/app-defaults/accessx.uid** before starting **accessx.**

FILES

accessx.uid	- Motif GUI description file
AccessX	- Application Resource File

SEE ALSO

X11(7)

AUTHORS

Earl Johnson
Jordan M. Slott
Enabling Technologies

Sun Microsystems Laboratories, Inc
Mountain View, CA

Mark Novak
Trace Research & Development Center
University of Wisconsin-Madison
Madison, WI

Will Walker
Digital Equipment Corportation
Maynard, MA

NAME

appres – list application resource database

SYNOPSIS

appres [[**class** [instance]] [**-1**] [*toolkitoptions...*]

DESCRIPTION

The **appres** program prints the resources seen by an application (or subhierarchy of an application) with the specified *class* and *instance* names. It can be used to determine which resources a particular program will load. For example,

```
% appres XTerm
```

will list the resources that any **xterm**(1) program will load. If no application class is specified, the class *-AppResTest-* is used.

To match a particular instance name, specify an instance name explicitly after the class name, or use the normal Xt toolkit option. For example,

```
% appres XTerm myxterm
```

or

```
% appres XTerm -name myxterm
```

To list resources that match a subhierarchy of an application, specify hierarchical class and instance names. The number of class and instance components must be equal, and the instance name should not be specified with a toolkit option. For example,

```
% appres Xman.TopLevelShell.Form xman.topBox.form
```

will list the resources of widgets of **xman**(1) topBox hierarchy. To list just the resources matching a specific level in the hierarcy, use the –1 option. For example,

```
% appres XTerm.VT100 xterm.vt100 -1
```

will list the resources matching the **xterm**(1) vt100 widget.

SEE ALSO

X11(7) **xrdb**(1)**, listres**(1)

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium

NAME

bdftopcf – convert font from Bitmap Distribution Format to Portable Compiled Format

SYNOPSIS

bdftopcf [-option ...] font-file.bdf

DESCRIPTION

Bdftopcf is the release 5 font compiler. Fonts in Portable Compiled Format can be read by any architecture, although the file is structured to allow one particular architecture to read them directly without reformatting. This allows fast reading on the appropriate machine, but the files are still portable (but read more slowly) on other machines.

OPTIONS

–**p***n* Sets the font glyph padding. Each glyph in the font will have each scanline padded in to a multiple of *n* bytes, where *n* is 1, 2, 4 or 8.

–**u***n* Sets the font scanline unit. When the font bit order is different from the font byte order, the scanline unit *n* describes what unit of data (in bytes) are to be swapped; the unit *i* can be 1, 2 or 4 bytes.

–**m** Sets the font bit order to MSB (most significant bit) first. Bits for each glyph will be placed in this order; i.e. the left most bit on the screen will be in the highest valued bit in each unit.

–**l** Sets the font bit order to LSB (least significant bit) first. The left most bit on the screen will be in the lowest valued bit in each unit.

–**M** Sets the font byte order to MSB first. All multi-byte data in the file (metrics, bitmaps and everything else) will be written most significant byte first.

–**L** Sets the font byte order to LSB first. All multi-byte data in the file (metrics, bitmaps and everything else) will be written least significant byte first.

–**t** When this option is specified, *bdftopcf* will convert fonts into "terminal" fonts when possible. A terminal font has each glyph image padded to the same size; the X server can usually render these types of fonts more quickly.

–**i** This option inhibits the normal computation of ink metrics. When a font has glyph images which do not fill the bitmap image (i.e. the "on" pixels don’t extend to the edges of the metrics) *bdftopcf* computes the actual ink metrics and places them in the .pcf file; the –t option inhibits this behaviour.

–**o <output-file-name>**
By default *bdftopcf* writes the pcf file to standard output; this option gives the name of a file to be used instead.

SEE ALSO

X11(7)

COPYRIGHT

Copyright 1991, Massachusetts Institute of Technology.
See *X11(7)* for a full statement of rights and permissions.

AUTHOR

Keith Packard, MIT X Consortium

NAME

bdftosnf – BDF to SNF font compiler for X11

SYNOPSIS

bdftosnf [**-p#**] [**-u#**] [**-m**] [**-l**] [**-M**] [**-L**] [**-w**] [**-W**] [**-t**] [**-i**] [*bdf-file*]

DESCRIPTION

bdftosnf reads a Bitmap Distribution Format (BDF) font from the specified file (or from standard input if no file is specified) and writes an X11 server normal font (SNF) to standard output.

OPTIONS

–**p#**	Force the glyph padding to a specific number. The legal values are 1, 2, 4, and 8.
–**u#**	Force the scanline unit padding to a specific number. The legal values are 1, 2, and 4.
–**m**	Force the bit order to most significant bit first.
–**l**	Force the bit order to least significant bit first.
–**M**	Force the byte order to most significant bit first.
–**L**	Force the byte order to least significant bit first.
–**w**	Print warnings if the character bitmaps have bits set to one outside of their defined widths.
–**W**	Print warnings for characters with an encoding of -1; the default is to silently ignore such characters.
–**t**	Expand glyphs in "terminal-emulator" fonts to fill the bounding box.
–**i**	Don't compute correct ink metrics for "terminal-emulator" fonts.

WARNING

SNF is an older font file format that is replaced by the Portable Compiled Format (PCF). PCF is the preferred format for font files and one can use **bdftopcf**(1) to convert a BDF font file to a PCF font file.

SEE ALSO

bdftopcf(1)

NAME

bitmap, bmtoa, atobm – bitmap editor and converter utilities for the X Window System

SYNOPSIS

bitmap [*–options* . . .] [*filename*] [*basename*]

bmtoa [**–chars** . . .] [*filename*]

atobm [**–chars** *cc*] [**–name** *variable*] [**–xhot** *number*] [**–yhot** *number*] [*filename*]

DESCRIPTION

The **bitmap** program is a rudimentary tool for creating or editing rectangular images made up of 1's and 0's. Bitmaps are used in X for defining clipping regions, cursor shapes, icon shapes, and tile and stipple patterns.

The **bmtoa** and **atobm** filters convert **bitmap** files (FILE FORMAT) to and from ASCII strings. They are most commonly used to quickly print out bitmaps and to generate versions for including in text.

COMMAND LINE OPTIONS

The **bitmap** command supports the standard X Toolkit command line arguments (see **X11**(7)). The following additional arguments are supported as well.

–**size** *WIDTHxHEIGHT*
Specifies size of the grid in squares.

–**sw** *dimension*
Specifies the width of squares in pixels.

–**sh** *dimension*
Specifies the height of squares in pixels.

–**gt** *dimension*
Grid tolerance. If the square dimensions fall below the specified value, grid will be automatically turned off.

–**grid, +grid**
Turns on or off the grid lines.

–**axes, +axes**
Turns on or off the major axes.

–**dashed, +dashed**
Turns on or off dashing for the frame and grid lines.

–**stippled, +stippled**
Turns on or off stippling of highlighted squares.

–**proportional, +proportional**
Turns proportional mode on or off. If proportional mode is on, square width is equal to square height. If proportional mode is off, **bitmap** will use the smaller square dimension, if they were initially different.

–**dashes** *filename*
Specifies the bitmap to be used as a stipple for dashing.

–**stipple** *filename*
Specifies the bitmap to be used as a stipple for highlighting.

–hl *color*
Specifies the color used for highlighting.

–fr *color*
Specifies the color used for the frame and grid lines.

filename
Specifies the bitmap to be initially loaded into the program. If the file does not exist, **bitmap** will assume it is a new file.

basename
Specifies the basename to be used in the C code output file. If it is different than the basename in the working file, **bitmap** will change it when saving the file.

The **bmtoa** command accepts the following option:

–chars *cc*
This option specifies the pair of characters to use in the string version of the bitmap. The first character is used for 0 bits and the second character is used for 1 bits. The default is to use dashes (–) for 0's and number signs (#) for 1's.

The **atobm** command accepts the following options:

–chars *cc*
This option specifies the pair of characters to use when converting string bitmaps into arrays of numbers. The first character represents a 0 bit and the second character represents a 1 bit. The default is to use dashes (–) for 0's and number signs (#) for 1's.

–name *variable*
This option specifies the variable name to be used when writing out the bitmap file. The default is to use the basename of the *filename* command line argument or leave it blank if the standard input is read.

–xhot *number*
This option specifies the X coordinate of the hotspot. Only positive values are allowed. By default, no hotspot information is included.

–yhot *number*
This option specifies the Y coordinate of the hotspot. Only positive values are allowed. By default, no hotspot information is included.

USAGE

The **bitmap** command displays grid in which each square represents a single bit in the picture being edited. Actual size of the bitmap image, as it would appear normaly and inverted, can be obtained by pressing **Meta-I** key. You are free to move the image popup out of the way to continue editing. Pressing the left mouse button in the popup window or **Meta-I** again will remove the real size bitmap image.

If the bitmap is to be used for defining a cursor, one of the squares in the images may be designated as the hot spot. This determines where the cursor is actually pointing. For cursors with sharp tips (such as arrows or fingers), this is usually at the end of the tip; for symmetric cursors (such as crosses or bullseyes), this is usually at the center.

Bitmaps are stored as small C code fragments suitable for including in applications. They provide an array of bits as well as symbolic constants giving the width, height, and hot spot (if specified) that may be used in creating cursors, icons, and tiles.

EDITING

To edit a bitmap image simply click on one of the buttons with drawing commands (**Point, Curve, Line, Rectangle,** etc.) and move the pointer into the bitmap grid window. Press one of the buttons on your mouse and the appropriate action will take place. You can either set, clear or invert the grid squares. Setting a grid square corresponds to setting a bit in the bitmap image to 1. Clearing a grid square corresponds to setting a bit in the bitmap image to 0. Inverting a grid square corresponds to changing a bit in the bitmap image from 0 to 1 or 1 to 0, depending what its previous state was. The default behavior of mouse buttons is as specified below.

MouseButton1	Set
MouseButton2	Invert
MouseButton3	Clear
MouseButton4	Clear
MouseButton5	Clear

This default behavior can be changed by setting the button function resources. An example is provided below.

```
bitmap*button1Function: Set
bitmap*button2Function: Clear
bitmap*button3Function: Invert
etc.
```

The button function applies to all drawing commands, including copying, moving and pasting, flood filling and setting the hot spot.

DRAWING COMMANDS

Here is the list of drawing commands accessible through the buttons at the left side of the application’s window. Some commands can be aborted by pressing A inside the bitmap window, allowing the user to select different guiding points where applicable.

Clear
This command clears all bits in the bitmap image. The grid squares will be set to the background color. Pressing C inside the bitmap window has the same effect.

Set This command sets all bits in the bitmap image. The grid squares will be set to the foreground color. Pressing S inside the bitmap window has the same effect.

Invert
This command inverts all bits in the bitmap image. The grid squares will be inverted appropriately. Pressing I inside the bitmap window has the same effect.

Mark
This command is used to mark an area of the grid by dragging out a rectangular shape in the highlighting color. Once the area is marked, it can be operated on by a

number of commands (see **Up, Down, Left, Right, Rotate, Flip, Cut,** etc.) Only one marked area can be present at any time. If you attempt to mark another area, the old mark will vanish. The same effect can be achieved by pressing **Shift-MouseButton1** and dragging out a rectangle in the grid window. Pressing **Shift-MouseButton2** will mark the entire grid area.

Unmark
This command will cause the marked area to vanish. The same effect can be achieved by pressing **Shift-MouseButton3**.

Copy
This command is used to copy an area of the grid from one location to another. If there is no marked grid area displayed, **Copy** behaves just like **Mark** described above. Once there is a marked grid area displayed in the highlighting color, this command has two alternative behaviors. If you click a mouse button inside the marked area, you will be able to drag the rectangle that represents the marked area to the desired location. After you release the mouse button, the area will be copied. If you click outside the marked area, **Copy** will assume that you wish to mark a different region of the bitmap image, thus it will behave like **Mark** again.

Move
This command is used to move an area of the grid from one location to another. Its behavior resembles the behavior of **Copy** command, except that the marked area will be moved instead of copied.

Flip Horizontally
This command will flip the bitmap image with respect to the horizontal axes. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing F inside the bitmap window has the same effect.

Up This command moves the bitmap image one pixel up. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing UpArrow inside the bitmap window has the same effect.

Flip Vertically
This command will flip the bitmap image with respect to the vertical axes. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing V inside the bitmap window has the same effect.

Left
This command moves the bitmap image one pixel to the left. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing LeftArrow inside the bitmap window has the same effect.

Fold
This command will fold the bitmap image so that the opposite corners become adjacent. This is useful when creating bitmap images for tiling. Pressing F inside the bitmap window has the same effect.

Right
This command moves the bitmap image one pixel to the right. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing

RightArrow inside the bitmap window has the same effect.

Rotate Left
This command rotates the bitmap image 90 degrees to the left (counter clockwise.) If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing L inside the bitmap window has the same effect.

Down
This command moves the bitmap image one pixel down. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing DownArrow inside the bitmap window has the same effect.

Rotate Right
This command rotates the bitmap image 90 degrees to the right (clockwise.) If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing R inside the bitmap window has the same effect.

Point
This command will change the grid squares underneath the mouse pointer if a mouse button is being pressed down. If you drag the mouse button continuously, the line may not be continuous, depending on the speed of your system and frequency of mouse motion events.

Curve
This command will change the grid squares underneath the mouse pointer if a mouse button is being pressed down. If you drag the mouse button continuously, it will make sure that the line is continuous. If your system is slow or **bitmap** receives very few mouse motion events, it might behave quite strangely.

Line
This command will change the grid squares in a line between two squares. Once you press a mouse button in the grid window, **bitmap** will highlight the line from the square where the mouse button was initially pressed to the square where the mouse pointer is located. By releasing the mouse button you will cause the change to take effect, and the highlighted line will disappear.

Rectangle
This command will change the grid squares in a rectangle between two squares. Once you press a mouse button in the grid window, **bitmap** will highlight the rectangle from the square where the mouse button was initially pressed to the square where the mouse pointer is located. By releasing the mouse button you will cause the change to take effect, and the highlighted rectangle will disappear.

Filled Rectangle
This command is identical to **Rectangle**, except at the end the rectangle will be filled rather than outlined.

Circle
This command will change the grid squares in a circle between two squares. Once you press a mouse button in the grid window, **bitmap** will highlight the circle from the square where the mouse button was initially pressed to the square where the mouse pointer is located. By releasing the mouse button you will cause the change

to take effect, and the highlighted circle will disappear.

Filled Circle
This command is identical to **Circle**, except at the end the circle will be filled rather than outlined.

Flood Fill
This command will flood fill the connected area underneath the mouse pointer when you click on the desired square. Diagonally adjacent squares are not considered to be connected.

Set Hot Spot
This command designates one square in the grid as the hot spot if this bitmap image is to be used for defining a cursor. Pressing a mouse button in the desired square will cause a diamond shape to be displayed.

Clear Hot Spot
This command removes any designated hot spot from the bitmap image.

Undo
This command will undo the last executed command. It has depth one, that is, pressing **Undo** after **Undo** will undo itself.

FILE MENU

The File menu commands can be accessed by pressing the File button and selecting the appropriate menu entry, or by pressing Ctrl key with another key. These commands deal with files and global bitmap parameters, such as size, basename, filename etc.

New
This command will clear the editing area and prompt for the name of the new file to be edited. It will not load in the new file.

Load
This command is used to load a new bitmap file into the bitmap editor. If the current image has not been saved, user will be asked whether to save or ignore the changes. The editor can edit only one file at a time. If you need interactive editing, run a number of editors and use cut and paste mechanism as described below.

Insert
This command is used to insert a bitmap file into the image being currently edited. After being prompted for the filename, click inside the grid window and drag the outlined rectangle to the location where you want to insert the new file.

Save
This command will save the bitmap image. It will not prompt for the filename unless it is said to be <none>. If you leave the filename undesignated or –, the output will be piped to stdout.

Save As
This command will save the bitmap image after prompting for a new filename. It should be used if you want to change the filename.

Resize
This command is used to resize the editing area to the new number of pixels. The

size should be entered in the WIDTHxHEIGHT format. The information in the image being edited will not be lost unless the new size is smaller that the current image size. The editor was not designed to edit huge files.

Rescale
This command is used to rescale the editing area to the new width and height. The size should be entered in the WIDTHxHEIGHT format. It will not do antialiasing and information will be lost if you rescale to the smaller sizes. Feel free to add you own algorithms for better rescaling.

Filename
This command is used to change the filename without changing the basename nor saving the file. If you specify – for a filename, the output will be piped to stdout.

Basename
This command is used to change the basename, if a different one from the specified filename is desired.

Quit
This command will terminate the bitmap application. If the file was not saved, user will be prompted and asked whether to save the image or not. This command is preferred over killing the process.

EDIT MENU

The Edit menu commands can be accessed by pressing the Edit button and selecting the appropriate menu entry, or by pressing Meta key with another key. These commands deal with editing facilities such as grid, axes, zooming, cut and paste, etc.

Image
This command will display the image being edited and its inverse in its actual size in a separate window. The window can be moved away to continue with editing. Pressing the left mouse button in the image window will cause it to disappear from the screen.

Grid
This command controls the grid in the editing area. If the grid spacing is below the value specified by gridTolerance resource (8 by default), the grid will be automatically turned off. It can be enforced by explicitly activating this command.

Dashed
This command controls the stipple for drawing the grid lines. The stipple specified by dashes resource can be turned on or off by activating this command.

Axes
This command controls the highlighting of the main axes of the image being edited. The actual lines are not part of the image. They are provided to aid user when constructing symmetrical images, or whenever having the main axes highlighted helps your editing.

Stippled
This command controls the stippling of the highlighted areas of the bitmap image. The stipple specified by stipple resource can be turned on or off by activating this command.

Proportional
This command controls the proportional mode. If the proportional mode is on, width and height of all image squares are forced to be equal, regardless of the proportions of the bitmap window.

Zoom
This command controls the zoom mode. If there is a marked area of the image already displayed, bitmap will automatically zoom into it. Otherwise, user will have to highlight an area to be edited in the zoom mode and bitmap will automatically switch into it. One can use all the editing commands and other utilities in the zoom mode. When you zoom out, undo command will undo the whole zoom session.

Cut This commands cuts the contents of the highlighted image area into the internal cut and paste buffer.

Copy
This command copies the contents of the highlighted image area into the internal cut and paste buffer.

Paste
This command will check if there are any other bitmap applications with a highlighted image area, or if there is something in the internal cut and paste buffer and copy it to the image. To place the copied image, click in the editing window and drag the outlined image to the position where you want to place i, and then release the button.

CUT AND PASTE

Bitmap supports two cut and paste mechanisms; the internal cut and paste and the global X selection cut and paste. The internal cut and paste is used when executing copy and move drawing commands and also cut and copy commands from the edit menu. The global X selection cut and paste is used whenever there is a highlighted area of a bitmap image displayed anywhere on the screen. To copy a part of image from another bitmap editor simply highlight the desired area by using the Mark command or pressing the shift key and dragging the area with the left mouse button. When the selected area becomes highlighted, any other applications (such as xterm, etc.) that use primary selection will discard their selection values and unhighlight the appropriate information. Now, use the Paste command for the Edit menu or control mouse button to copy the selected part of image into another (or the same) bitmap application. If you attempt to do this without a visible highlighted image area, the bitmap will fall back to the internal cut and paste buffer and paste whatever was there stored at the moment.

WIDGETS

Below is the widget structure of the **bitmap** application. Indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name. All widgets except the bitmap widget are from the standard Athena widget set.

```
Bitmap bitmap
        TransientShell image
                Box box
                        Label normalImage
                Label invertedImage
        TransientShell input
            Dialog dialog
                Command okay
                Command cancel
        TransientShell error
            Dialog dialog
                Command abort
                Command retry
        TransientShell qsave
            Dialog dialog
                Command yes
                Command no
                Command cancel
        Paned parent
            Form formy
                MenuButton fileButton
                SimpleMenu fileMenu
                    SmeBSB  new
                    SmeBSB  load
                    SmeBSB  insert
                    SmeBSB  save
                    SmeBSB  saveAs
                    SmeBSB  resize
                    SmeBSB  rescale
                    SmeBSB  filename
                    SmeBSB  basename
                    SmeLine line
                    SmeBSB  quit
                MenuButton editButton
                SimpleMenu editMenu
                    SmeBSB  image
                    SmeBSB  grid
                    SmeBSB  dashed
                    SmeBSB  axes
                    SmeBSB  stippled
                    SmeBSB  proportional
                    SmeBSB  zoom
                    SmeLine line
                    SmeBSB  cut
                    SmeBSB  copy
                    SmeBSB  paste
                Label status
            Pane pane
                Bitmap bitmap
```

```
                                   Form form
                                           Command clear
                                           Command set
                                           Command invert
                                           Toggle  mark
                                           Command unmark
                                           Toggle  copy
                                           Toggle  move
                                           Command flipHoriz
                                           Command up
                                           Command flipVert
                                           Command left
                                           Command fold
                                           Command right
                                           Command rotateLeft
                                           Command down
                                           Command rotateRight
                                           Toggle  point
                                           Toggle  curve
                                           Toggle  line
                                           Toggle  rectangle
                                           Toggle  filledRectangle
                                           Toggle  circle
                                           Toggle  filledCircle
                                           Toggle  floodFill
                                           Toggle  setHotSpot
                                           Command clearHotSpot
                                           Command undo
```

COLORS

If you would like bitmap to be viewable in color, include the following in the #ifdef COLOR section of the file you read with xrdb:

```
*customization:              –color
```

This will cause bitmap to pick up the colors in the app-defaults color customization file: /usr/openwin/lib/app-defaults/Bitmap.

BITMAP WIDGET

Bitmap widget is a stand-alone widget for editing raster images. It is not designed to edit large images, although it may be used in that purpose as well. It can be freely incorporated with other applications and used as a standard editing tool. The following are the resources provided by the bitmap widget.

Bitmap Widget

Header file	Bitmap.h
Class	bitmapWidgetClass
Class Name	Bitmap
Superclass	Bitmap

All the Simple Widget resources plus . . .

Name	**Class**	**Type**	**Default Value**
foreground	Foreground	Pixel	XtDefaultForeground
highlight	Highlight	Pixel	XtDefaultForeground
framing	Framing	Pixel	XtDefaultForeground
gridTolerance	GridTolerance	Dimension	8
size	Size	String	32x32
dashed	Dashed	Boolean	True
grid	Grid	Boolean	True
stippled	Stippled	Boolean	True
proportional	Proportional	Boolean	True
axes	Axes	Boolean	False
squareWidth	SquareWidth	Dimension	16
squareHeight	SquareHeight	Dimension	16
margin	Margin	Dimension	16
xHot	XHot	Position	NotSet (–1)
yHot	YHot	Position	NotSet (–1)
button1Function	Button1Function	DrawingFunction	Set
button2Function	Button2Function	DrawingFunction	Invert
button3Function	Button3Function	DrawingFunction	Clear
button4Function	Button4Function	DrawingFunction	Invert
button5Function	Button5Function	DrawingFunction	Invert
filename	Filename	String	None ("")
basename	Basename	String	None ("")

AUTHOR

Davor Matic, X Consortium

NAME

bldrgb – simple interface to the rgb(1) utility

SYNOPSIS

bldrgb

DESCRIPTION

bldrgb is a simple interface to the **rgb** utility which is used to build a color name database. **bldrgb** searches for the *rgb.txt* color name input file and then executes the **rgb** command, passing it the appropriate arguments.

EXAMPLES

```
example% bldrgb
```

FILES

/usr/openwin/lib/rgb.txt	color name database source. Maps color names to RGB color values.
/usr/openwin/lib/rgb.dir	dbm file containing color name to RGB mapping.
/usr/openwin/lib/rgb.pag	dbm file containing the color name to RGB mapping.

SEE ALSO

cat(1), **rgb**(1), **showrgb**(1), **dbm**(3)

NAME

cmap_alloc – allocate default colormaps for non-default visuals

SYNOPSIS

cmap_alloc [–**display** *display:n.screen*] [–**force**] [–**allscreens**]
[–**depth** *n*] [–**visual** *class*] [–**verbose**] [–**help**]

DESCRIPTION

cmap_alloc creates empty colormaps for non-default dynamic visuals that are suitable for use as default colormaps. An X11 server has only one default colormap, and it is associated with the default visual. Clients that need to access non-default visuals have to create their own colormaps. This can lead to gratuitous colormap flashing if a lot of applications are running in non-default visuals. **cmap_alloc** will create colormaps that can be used by such applications as the default colormap for the given visual. Once the colormap has been created for a given visual, it will place the colormap's XID into the RGB_DEFAULT_MAP property on the root window of the display. Clients that need to access non-default dynamic visuals should search this property using XGetRBGColormaps() for a suitable colormap to use.

OPTIONS

–**display** *display:n.screen*
: Indicates which X11 server to use. The default is to use the value set in the DISPLAY environment variable.

–**force**
: Create a colormap even if a suitable colormap is already listed in the RGB_DEFAULT_MAP property.

–**allscreens**
: Create default colormaps for all screens of the display. Without this option, colormaps will only be created for the default screen.

–**depth** *n*
: Create default colormaps only for those visuals that have depth *n*. The default is to create colormaps for visuals of all depths

–**visual** *class*
: Create default colormaps only for those visuals that are of class *class*. Where *class* is one of the following: GrayScale, PsuedoColor, DirectColor

–**verbose**
: Print out diagnostic messages useful for debugging.

–**help**
: Print out a short usage message and exit.

SEE ALSO

xstdcmap(1), **cmap_compact**(1)

DIAGNOSTICS

can't open display ...
: Error in connecting to the X11 server. Check DISPLAY environment variable.

unknown depth
: Invalid depth specified for the –**depth** command line option

unknown visual class
: Invalid visual class specified for the –**visual** command line option

NAME

cmap_compact – colormap configuration utility to reduce colormap flashing.

SYNOPSIS

cmap_compact [*op*] [**–display** *dpyname*]

DESCRIPTION

cmap_compact is a utility which allows certain colors to be designated as special and to be promoted to the high end of the default colormap (toward pixel 255). This reduces colormap flashing in many situations.

This technique has no effect for monochrome screens; it applies only to color screens.

For more information about the general problem of colormap flashing and instructions for using **cmap_compact** to solve the problem, refer to the OpenWindows documentation.

OPTIONS

-display *dpyname*
    Indicates the X11 display to use. ':0' is the default.

op can be one of the following operations:

save Records the RGB color values of all shareable (read-only) colors in the default colormap for each active screen and saves these values on the file **~/.owcolors**.

The black and white pixels (typically at pixels 254 and 255) are ignored during the save.

Note that all colors are ignored for screens with a StaticColor default colormap. Note also that the server is grabbed during the save.

init Reads the colors which have been saved in **~/.owcolors** and allocates them as read-only colors at the high-end of the default colormap. The color allocations for those applications which use these colors will be derived from these high colors. If there is no **~/.owcolors** file, nothing happens and no message is printed out.

The colors are allocated in reverse order of the order in which they were saved. Thus, the lowest pixel in the colormap at save time becomes the highest pixel at initialization time.

This operation is typically invoked from **~/.xinitrc** prior to starting the color applications.

This operation creates on the root window of each screen a property named *XA_COMPACTED_COLORS_COUNT* of type XA_INTEGER. The value of this property specifies the number of initialized colors for that screen. Note: this value also includes the black and white pixels.

An X11 *close down mode* of **RetainPermanent** is set so that the colors this program allocates will stay around after **cmap_compact** exits. These colors may be freed by calling **cmap_compact dealloc** . Before this program allocates its colors, it frees any colors left over from a previous invocation.

If there are more saved colors than can fit in the default colormap, as many colors as will fit are allocated and a warning message is printed out. This situation, for example, might occur if the the default visual were switched from an 8-bit to a 4-bit visual without discarding saved colors.

Note: no allocations occur for screens with a StaticColor default colormap.
Note: The server is grabbed during the initialization.

discard
This operation removes the file **~/.owcolors**.

dealloc This operation frees colors allocated by a previous invocation of **cmap_compact init**.

show This operation prints out the RGB values of the saved colors for each screen.

FILES

~/.owcolors
The file which is generated by **cmap_compact save**. Contains a list of saved, read-only workspace colors for all screens.

SEE ALSO

Xsun(1), **openwin**(1)
Solaris X Window System Developers Guide

BUGS

Currently does not support using the same **~/.owcolors** file for different machine architectures.

NAME

constype – print type of Sun console

SYNOPSIS

constype

DESCRIPTION

constype prints on the standard output the Sun code for the type of display that the console is. It is one of:

bw?	Black and White, where ? is 1-4. (eg) 3-50s are bw2
cg?	Colour Graphics display, where ? is 1-4
gp?	Optional Graphics Processor board, where ? is 1-2
gx	Sun GX (cg6) Graphics Accelerator
gs	Sun GS (cg12) Graphics Accelerator
gt	Sun GT Graphics Accelerator
ns?	Not Sun display - where ? is A-J

This is useful in determining startup values and defaults for window systems.

BUGS

Not tested on all monitor types

COPYRIGHT

Copyright 1988, SRI

AUTHOR

Doug Moran <moran@ai.sri.com>

NAME

cps – construct C to POSTSCRIPT language interface

SYNOPSIS

cps [–**c** [%] | –**C** [%]] [–**o** *style*] [–**f** *filespec*] [–**D** *symbol*] [–**I** *filename*] [-**s** [*number*]]
[–**i**] [–**l** [**on** | **off**]] [*file*.**cps**]

DESCRIPTION

cps compiles a specification file containing C procedure names and POSTSCRIPT language code into a header file (*file*.**h**) that can be included in C programs. The header file associates the C procedure names with macros that transmit a compressed form of the POSTSCRIPT language code to the X11 / NeWS server stream. Only one input file can be specified. If the *file*.**{c,h,cc}** file(s) already exist, a backup will be generated of the form *file*.**{c,h,cc}.BAK** before the new file is generated.

The convention is for the input specification file to end with the suffix **.cps**.

OPTIONS

–**c** Compiles the file of POSTSCRIPT language code for faster loading by NeWS, but does not generate a specification file for C programs. For example, the command line

example% cps -c < input_file > output_file

converts the input file from the ASCII form of the POSTSCRIPT language to a compressed binary form. When read by NeWS, the output file will execute exactly the same as *input_file*, except that it will be parsed faster.

NOTE: The –**c** option will not work if the input file uses constructs such as **currentfile readstring**, which are often used with the **image** primitive.

–**c%** Identical with **-**c, but will copy without compression until the first percent (%) character is found. This is to prevent cps from modifying shell commands in a file which is intended to be directly executed.

–**C** Compiles the file of POSTSCRIPT language code in the same way that the **-c** option does: however, when **-C** is used, the file can contain usertoken specifications. The tokens are automatically set up at the start of the output file; they are used throughout the output file to compress the POSTSCRIPT language even further than occurs with the **-c** option.

–**C%** Identical with **-**C, but will copy without compression until the first percent (%) character is found. This is to prevent cps from modifying shell commands in a file which is intended to be directly executed.

–**D** *symbol* Defines symbols to be passed to the C language preprocessor (**cpp**(1)), which processes the input file.

–**f** *filespec* Indicates the connection file specification. In an environment where there may be multiple PSFILE or wire connections open it is convienient to be able to directly specify which file is to be used to send / recieve the function data. (The examples below assume the ansi output option is in effect.) Valid specifications are:

current Generates code which uses the global PostScript and PostScriptInput variables. "cdef mumble(int foo)" becomes "...

mumble (int foo)"

wire Generates code which expects a wire to be passed to the cdef'd functions / macros. "cdef mumble (int foo)" becomes "... mumble (wire_Wire The__Wire, int foo)"

psfile Generates code which expects the input side of a PSFILE pair to be passed to the cdef'd functions / macros. "cdef mumble (int foo)" becomes "... mumble (PSFILE ∗The__File, int foo)"

–**I** *filename* Specifies include files or include paths. Passed on to the C preprocessor.

–**o** *style* Specifies the output format.

Most of these options are useful for minimizing the size impact of CPS interfaces that contain procedures called from several places in the C code. The *file*.**c** generated would only need to be compiled once. Each file that needs to use the interface could then include only the *file*.**h** and use the macros in that file multiple times. Each repeated invocation of the macro would refer to the shared POSTSCRIPT language code in the *file*.**c** rather than its own static copy of the POSTSCRIPT language code.

In all cases if the output file(s) exist, **.BAK** backup copies will be generated.

styles recognized are:

macros Generates a single specification file of the form file.h, which contains the string constants and macro definitions required for the C-POSTSCRIPT interface.

ansi Generates two specification files: the first file contains only ANSI-c function prototypes; the second file contains the function definitions required for the C-POSTSCRIPT interface. The first file is of the form *file*.**c**; the second file is of the form *file*.**h**.

c++ Generates code identical to the ansi option, but the second file is of the form file.cc.

conststr Generates two specification files: the first file contains only the compressed form of the POSTSCRIPT language code; the second file contains the macro definitions required for the C-POSTSCRIPT interface. For example, **ps_open_PostScript()** and **ps_close_PostScript()** would be defined in the second file. The second file references compressed POSTSCRIPT language code as **extern char** arrays. The first file is of the form *file*.**c**; the second file is of the form *file*.**h**.

functions Generates two specification files: the first file contains only traditional (K&R) c function prototypes; the second file contains the function definitions required for the C-POSTSCRIPT interface. The first file is of the form *file*.**c**; the second file is of the form *file*.**h**.

inline Generates a single specification file of the form file.h, which contains the function definitions required for the C-POSTSCRIPT interface expressed as ANSI static inline functions.

portable Generates two specification files: the first file contains only function prototypes declared using the extended portability macros defined in <portable/c_varieties.h>. The second file contains the traditional (K&R) function definitions required for the C-POSTSCRIPT interface. The first file is of the form *file*.**c**; the second file is of the form *file*.**h**.

–i Obsolete. Replaced by '-o conststr'.

-s [*number*]
Specifies the threshold at which compiled POSTSCRIPT language code will be output as decimal arrays instead of string constants. If *number* is missing, all POSTSCRIPT language code will be ouput as string constants in the resulting *file*.**h**. This may be useful for debugging purposes, even though the POSTSCRIPT language code is in compressed form. If *number* is 0, all POSTSCRIPT language code will be output as decimal arrays. The default threshold is 400 characters, which is less than the maximum limit of string constants for most compilers. Note that there must be no space before *number*, since it is optional.

–l[on | off]
Specify the handling of '(literal string) gettext' sequences in cdef functions. When turned on ('-l' or '-lon'), the c code generated will perform a gettext() on the literal string before sending it down the wire to the NeWS server; and the 'gettext' token won't be sent at all. The default is '-loff'.

SEE ALSO

cpp(1)
NeWS 3.1 Programmer's Guide, Appendix B - Byte Stream

TRADEMARK

POSTSCRIPT is a registered trademark of Adobe Systems, Inc.

NAME

dps – Display PostScript imaging for the X Window System

DESCRIPTION

This manual page provides information about the Display PostScript system, implemented as an extension to the X Window System.

The INTRODUCTION section contains a brief, nontechnical description of the Display PostScript system.

The remaining sections provide the application developer with more detailed technical information about the architecture.

The REFERENCES section describes additional documentation and tells you how to use Adobe's public access file server.

INTRODUCTION

The PostScript language is a simple interpretive programming language with powerful graphics capabilities. Its primary purpose is to describe the appearance of text, graphical shapes, and images on printed or displayed pages. If an application, such as a word processing system or graphics package, produces a page description using the PostScript language, you can print the pages on a wide variety of PostScript printers and view them on monitors where the Display PostScript system is available.

The Display PostScript system is a high-performance implementation of the PostScript language for interactive computer displays. The use of the Display PostScript system ensures true WYSIWYG (What You See Is What You Get) between the display and any PostScript printer.

DISPLAY POSTSCRIPT SYSTEM ARCHITECTURE

The Display PostScript system is part of the X Window System and is implemented as an X extension. Display PostScript applications use window system features for window placement and sizing, menu creation, and event handling, while using Display PostScript features to take care of imaging inside the window.

Display PostScript system components include:

The PostScript interpreter.

The Client Library – a C language interface to the basic facilities of the Display PostScript system.

pswrap – a preprocessor that prepares PostScript language programs for invocation from a C program.

These components are discussed below.

APPLICATION BUILDING BLOCKS

Most of a Display PostScript application is written in C or another high-level language. It calls Client Library procedures to start a PostScript execution context, send programs and data to the PostScript interpreter, and get results from the interpreter. The Client Library is the application's primary interface to the Display PostScript system.

In addition, it calls *wraps* – custom PostScript language procedures developed specifically for the application. Wraps are generated by the *pswrap* translator from application-specific PostScript language code.

USING PSWRAP

pswrap is a preprocessor that takes PostScript language code as input and embeds it in C-callable procedures, or wraps. The output of *pswrap* is compiled and linked with the rest of your application, which can then call the wraps to transmit PostScript language code to the PostScript interpreter.

A Display PostScript application uses C or another high-level language to perform calculations, communicate with the window system, read and write files, and do other application processing. It uses wraps primarily for imaging tasks.

Consider a procedure, **PSWDisplayText**, that places text on the screen at a particular *x,y* coordinate. A call to this wrap from the application program might look something like this:

```
PSWDisplayText(72.0, 100.0, "Hello World");
```

The body of the **PSWDisplayText** procedure is actually written in the PostScript language. It was defined to *pswrap* as follows:

```
defineps PSWDisplayText(float X,Y; char *text)
        X Y moveto
        (text) show
endps
```

In the wrap definition, the **defineps** and **endps** keywords tell *pswrap* where a given PostScript language program begins and ends. The **defineps** statement defines the resulting procedure call. The *pswrap* translator processes this input and produces a C language source-code file. When compiled and linked with the application, the **PSWDisplayText** procedure sends a PostScript language program to the interpreter (binary-encoded for more efficient processing), causing "Hello World" to be displayed on the screen.

See the *Programming the Display PostScript System with X* for further information.

THE CLIENT LIBRARY

The Display PostScript Client Library is a linkable library of compiled C procedures that provides an interface between the application and the Display PostScript system. It creates an environment for handling imaging calls to specific Client Library procedures like **DPSmoveto** and to custom wraps written for the application.

To the application programmer, it appears that Client Library procedures directly produce graphical output on the display. In fact, these procedures generate PostScript language statements and transmit them to the PostScript interpreter for execution; the PostScript interpreter then produces graphical output that is displayed by device-specific procedures in the Display PostScript system. In this way, the Client Library makes the full power of the PostScript interpreter and imaging model available to a C language program.

The Client Library includes procedures for creating, communicating with, and destroying PostScript execution contexts. A context consists of all the information (or "state") needed by the PostScript interpreter to execute a PostScript language program. In the Client

Library interface, each context is represented by a **DPSContextRec** data structure. PostScript execution contexts are described in the *PostScript Language Reference Manual, Second Edition*.

REFERENCES

Information about the PostScript Language and the Display PostScript system is available in a number of manuals and via the public access file server described below.

POSTSCRIPT LANGUAGE MANUALS

If you're new to the PostScript language, you should first read the following manuals (published by Addison-Wesley and available from Adobe Systems Incorporated or through your technical bookstore):

PostScript Language Reference Manual, Second Edition

The standard reference for the PostScript language. Describes the PostScript imaging model and the concepts and facilities of the PostScript interpreter. Documents the PostScript language. Required reading.

PostScript Language Tutorial and Cookbook

Introduction to the PostScript language in an informal, interactive style. Contains a collection of example programs that illustrate the PostScript imaging model.

PostScript Language Program Design

Guidelines for the advanced developer to use in designing and debugging PostScript language programs. Printer-oriented, but most of the information is relevant to writing a Display PostScript application.

DISPLAY POSTSCRIPT MANUALS

Once you're up to speed in the PostScript language, read *Programming the Display PostScript System with X*, available from Addison-Wesley. This book is collection of manuals that explain how to render text and graphics with the Display PostScript extension to X. It contains the following manuals:

Programming Guide

Explains how to render text and graphics with the Display PostScript extension to X.

Client Library Reference Manual

Describes the procedural interface to the Display PostScript system. Tells how to send programs and data to a PostScript execution context, how to handle context output, how to create and terminate a context. Contains procedure definitions, programming tips, and a sample application program.

Client Library Supplement for X

Describes Display PostScript features that are specific to the X Window System, such as context creation and additional error codes.

pswrap Reference Manual

Describes how to define C-callable procedures that contain PostScript language programs. Tells how to declare input arguments and output to be received from the interpreter. Documents the pswrap command line options.

Display PostScript Toolkit for X

Describes the Display PostScript Toolkit for the X Window System. It also contains information about locating PostScript language resources and about the *makepsres* utility.

THE PUBLIC ACCESS FILE SERVER

Adobe Systems Incorporated provides a public access file server. If you have access to Internet or UUCP electronic mail, you can use the public access file server to obtain the following information:

Display PostScript system manuals

Code examples

AFM files

Documentation updates

The public access file server is a mail-response program. That is, you send it a request by electronic mail and it mails back a response. (The ‘‘Subject:’’ line is treated as part of the message by the file server.)

To send mail to the file server, use one of the following addresses:

Internet	**ps-file-server@adobe.com**
UUCP	**...!decwrl!adobe!ps-file-server**

To receive a quick summary of file server commands, send the following message:

help

To receive detailed information on how to use the file server, send the following message:

send Documents long.help

COLORMAP USAGE

The Display PostScript system uses entries from the default X colormap to display colors and grey values. You can configure this usage. Giving the Display PostScript system more colormap entries improves the quality of its rendering, but leaves fewer entries available to other applications since the default colormap is shared.

Resources in your .Xdefaults file control the colormap usage. Each resource entry should be of the form

DPSColorCube.visualType.depth.color: size

where

visualType is one of GrayScale, PseudoColor, or DirectColor.

depth is 1, 2, 4, 8, 12, or 24 and should be the largest depth equal to or less than the default depth.

color is one of the strings "reds", "greens", "blues", or "grays".

size is the number of values to allocate of that color.

These resources are not used for the static visual types StaticGray, StaticColor, or TrueColor. Specifying 0 for reds directs the Client Library to use only a gray ramp. This specification is particularly useful for gray-scale systems that incorrectly have

PseudoColor as the default visual.

For example, to configure a 5x5x4 color cube and a 17-element gray ramp for an 8-bit PseudoColor screen, specify these resources:

```
DPSColorCube.PseudoColor.8.reds: 5
DPSColorCube.PseudoColor.8.greens: 5
DPSColorCube.PseudoColor.8.blues: 4
DPSColorCube.PseudoColor.8.grays: 17
```

These resources use 117 colormap entries, 100 for the color cube and 17 for the gray ramp. For the best rendering results, specify an odd number for the gray ramp.

Resources that are not specified take these default values:

```
DPSColorCube.GrayScale.4.grays: 9
DPSColorCube.GrayScale.8.grays: 17

DPSColorCube.PseudoColor.4.reds: 2
DPSColorCube.PseudoColor.4.greens: 2
DPSColorCube.PseudoColor.4.blues: 2
DPSColorCube.PseudoColor.4.grays: 2
DPSColorCube.PseudoColor.8.reds: 4
DPSColorCube.PseudoColor.8.greens: 4
DPSColorCube.PseudoColor.8.blues: 4
DPSColorCube.PseudoColor.8.grays: 9
DPSColorCube.PseudoColor.12.reds: 6
DPSColorCube.PseudoColor.12.greens: 6
DPSColorCube.PseudoColor.12.blues: 5
DPSColorCube.PseudoColor.12.grays: 17

DPSColorCube.DirectColor.12.reds: 6
DPSColorCube.DirectColor.12.greens: 6
DPSColorCube.DirectColor.12.blues: 6
DPSColorCube.DirectColor.12.grays: 6
DPSColorCube.DirectColor.24.reds: 7
DPSColorCube.DirectColor.24.greens: 7
DPSColorCube.DirectColor.24.blues: 7
DPSColorCube.DirectColor.24.grays: 7
```

If none of the above defaults apply to the display, the Client Library uses no color cube and a 2-element gray ramp; that is, black and white.

SEE ALSO

pswrap(1), **dpsexec**(6)

NOTES

Copyright 1988-1992 Adobe Systems Incorporated.

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

NAME

dpsexec – Display PostScript Executive

SYNOPSIS

dpsexec [–**display** *name*][–**sync**][–**backup**][–**noexec**]

DESCRIPTION

dpsexec is a Display PostScript program that allows the user to interact directly with the PostScript interpreter through a command interface. **dpsexec** reads lines of text from standard input and passes each line to the PostScript interpreter for execution. It creates a window that displays the results of graphics operations as they are executed. **dpsexec** exits when end of file is reached on standard input, or when the user types "quit<return>", which executes the PostScript **quit** operator.

By default, **dpsexec** executes the PostScript **executive** operator before it accepts any user input. This operator puts the PostScript interpreter in "interactive executive" mode so that the user can control the interpreter directly. In this mode, the PostScript interpreter supports certain line-editing functions and prompts the user when it is ready to execute more input. See section 2.4.4, "Using the Interpreter Interactively," of the *PostScript Language Reference Manual, Second Edition*, for detailed information on this mode of operation.

OPTIONS

–**display** *name*
Specifies the display on which to open a connection to the Display PostScript system. If no display is specified, the DISPLAY environment variable is used.

–**sync** Establishes a synchronous connection with the specified X display.

–**backup**
Uses backing store for the window in which graphics are displayed, if possible.

–**noexec**
Prevents **dpsexec** from entering "interactive executive" mode. The primary effect of this option is to inhibit printing the **PS>** prompt before each line of input is accepted. This option is useful when **dpsexec** is run with standard input redirected from a file or a pipe.

DIAGNOSTICS

PostScript language error messages are printed to standard output.

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

NAME

editres – a dynamic resource editor for X Toolkit applications

SYNTAX

editres [-*toolkitoptions...*]

OPTIONS

Editres accepts all of the standard X Toolkit command line options (see **X11**(7)). The order of the command line options is not important.

DESCRIPTION

Editres is a tool that allows users and application developers to view the full widget hierarchy of any X Toolkit client that speaks the **editres** protocol. In addition **editres** will help the user construct resource specifications, allow the user to apply the resource to the application and view the results dynamically. Once the user is happy with a resource specification editres will append the resource string to the user's X Resources file.

USING EDITRES

Editres provides a window consisting of the following four areas:

Menu Bar	A set of popup menus that allow you full access to editres's features.
Panner	The panner allows a more intuitive way to scroll the application tree display.
Message Area	Displays information to the user about the action that editres expects of the her.
Application Widget Tree	This area will be used to display the selected client's widget tree.

To begin an editres session select the **Get Widget Tree** menu item from the command menu. This will change the pointer cursor to cross hair. You should now select the application you wish look at by clicking on any of its windows. If this application understands the editres protocol then editres will display the client's widget tree in its tree window. If the application does not understand the editres protocol editres will inform you of this fact in the message area after a few seconds delay.

Once you have a widget tree you may now select any of the other menu options. The effect of each of these is described below.

COMMANDS

Send Widget Tree

Allows the user to click on any client that speaks the editres protocol and receive its widget tree.

Refresh Widget Tree

Editres only knows about the widgets that exist at the present time. Many applications create and destroy widgets "on-the-fly". Selecting this menu item will cause editres to ask the application to resend its widget tree, thus updating its information to the new state of the application.

Example

Xman only creates the widgets for its *topbox* when it starts up. None of the widgets for the manual page window are created until the user actually clicks on the *Manual Page* button. If you retrieved xman's widget tree before the the manual page is active, you may wish to refresh the widget tree after the manual page has been displayed. This will allow you to also edit the manual page's resources.

Select Widget in Client
: This menu item allows you to select any widget in the application, editres will then highlight the corrosponding element the widget tree display. Once this menu item is selected the pointer cursor will again turn to a crosshair, and you must click any pointer button in the widget you wish to have displayed. Since some widgets are fully obscured by their children, it is not possible to get to every widget this way, but this mechanism does give very useful feedback between the elements in the widget tree and those in the actual client.

Dump Widget Tree to a File
: For documenting applications it is often useful to be able to dump the entire application widget tree to an ascii file. This file can then be included in the manual page. When this menu item is selected a popup dialog is activated. Type the name of the file in this dialog, and either select *okay*, or type a carriage-return. Editres will now dump the widget tree to this file. To cancel the file dialog just select the *cancel* button.

Show Active Widgets
: This command is the inverse of the **Select Widget in Client** command, it will show the user each widget that is currently selected in the widget tree, by flashing the corresponding widget in the application *numFlashes* (three by default) times in the *flashColor*.

Show Resource Box
: This command will popup a resource box for the current client. This resource box (described in detail below) will allow the user to see exactly which resources can be set for the widget that is currently selected in the widget tree display. Only one widget may be currently selected, if greater or fewer are selected editres will refuse to pop up the resource box, and put an error message in the **Message Area**.

Show Author
: Pops up message telling you who wrote this application.

Quit
: Exits editres.

TREE COMMANDS

The **Tree** menu contains several commands that allow operations to be performed on the widget tree.

Select All
Unselect All
Invert All
: These functions allow the user to select, unselect, or invert all widgets in the widget tree.

Select Children
Select Parents

These functions select the immediate parent or children of each of the currently selected widgets.

Select Descendants
Select Ancestors
These functions select all parents or children of each of the currently selected widgets. This is a recursive search.

Show Widget Names
Show Class Names
Show Widget Windows
When the tree widget is initially displayed the labels of each widget in the tree correspond to the widget names. These functions will cause the label of **all** widgets in the tree to be changed to show the class name, IDs, or window associated with each widget in the application. The widget IDs, and windows are shown as hex numbers.

In addition there are keyboard accelerators for each of the Tree operations. If the input focus is over an individual widget in the tree, then that operation will only effect that widget. If the input focus is in the Tree background it will have exactly the same effect as the corresponding menu item.

The translation entries shown may be applied to any widget in the application. If that widget is a child of the Tree widget, then it will only affect that widget, otherwise it will have the same effect as the commands in the tree menu.

Key	**Option**	**Translation Entry**
space	Unselect	Select(nothing)
s	Select	Select(all)
i	Invert	Select(invert)
c	Select Children	Select(children)
d	Select Descendants	Select(descendants)
p	Select Parent	Select(parent)
a	Select Ancestors	Select(ancestors)
N	Show Widget Names	Relabel(name)
C	Show Class Names	Relabel(class)
I	Show Widget IDs	Relabel(id)
W	Show Widget Windows	Relabel(window)

USING THE RESOURCE BOX

The resource box contains five different areas. Each of the areas, as they appear on the screen, from top to bottom will be discussed.

The Resource Line
This area at the top of the resource box shows the current resource entry exactly as it would appear if you were to save it to a file.

The Widget Names and Classes
This area allows you to select exactly which widgets this resource will apply to.

The area contains three lines, the first contains the name of the selected widget and all its ancestors, and the more restrictive dot (**.**) separator. The second line contains less specific the Class names of each widget, and well as the less restrictive star (*) separator. The last line contains a set of special buttons called **Allow Any Widget** which will completely compress out this level of the widget hierarchy, replacing it with just the star separator.

The initial state of this area is the most restrictive, using the resource names and the dot separator. By selecting the other buttons in this are you can ease the restrictions to allow more and more widgets to match the specification. The extreme case is to select all the **Allow Any Widget** buttons, which will match every widget in the application. As you select different buttons the tree display will update to show you exactly which widgets will be effected by the current resource specification.

Normal and Constrain Resources
: The next area allows you to select the name of the normal or constraint resources you wish to set. Some widgets may not have constraint resources, so that area will not appear.

Resource Value
: This next area allows you to enter the resource value. This value should be entered exactly as you would type a line into your resource file. Thus it should contain no unescaped new-lines. There are a few special character sequences for this file:

\n - This will be replaced with a newline.

\### - Where # is any octal digit. This will be replaced with a single byte that contains this sequence interpreted as an octal number. For example, a value containing a NULL byte can be stored by specifying \000.

\<new-line> - This will compress to nothing.

\\ - This will compress to a single backslash.

Command Area
: This area contains several command buttons that I will describe in this section.

Set Save File
: This button allows the user to modify file that the resources will be saved to. This button will bring up a dialog box that will ask you for a filename, once the filename has been entered, either hit carriage-return or click on the *okay* button. To popdown the dialog box without changing the save file, click the *cancel* button.

Save
: This button will append the **resource line** described above to the end of the current save file. If no save file has been set the **Set Save File** dialog box will be popped up to prompt the user for a filename.

Apply
: This button attempts to perform a XtSetValues call on all widgets that match the

resource line described above. The value specified is applied directly to all matching widgets. This behavior is an attempt to give a dynamic feel to the resource editor. Since this feature allows users to put an application in states it may not be willing to handle, a hook has been provided to allow specific clients block these SetValues requests (see **Blocking Editres Requests** below).

Unfortunately due to design constraints imposed on the widgets by the X Toolkit and the Resource Manager, trying to coerce an inherently static system into dynamic behavior can cause strange results. There is no guarantee that the results of an apply will be the same as what will happen when you save the value, and restart the application. This functionality is provided to try to give you a rough feel for what your changes will accomplish, and the results obtained should be considered suspect at best. Having said that, this is one of the neatest features of editres, and I strongly suggest that you play with it, and see what it can do.

Save and Apply
This button combines the Save and Apply actions described above into one button.

Popdown Resource Box
This button will remove the resource box from the display.

BLOCKING EDITRES REQUESTS

The editres protocol has been built into the Athena Widget set. This allows all application that are linked against Xaw to be able to speak to the resource editor. While this provides great flexability, and is a useful tool, it can quite easily be abused. It is therefore possible for any Xaw client to specify a value for the **editresBlock** resource described below, to keep editres from divulging information about its internals, or to disable the **SetValues** part of the protocol.

editresBlock (Class **EditresBlock)**
Specifies which type of blocking this client wishes to impose on the editres protocol.

The accepted values are:

all	Block all requests.
setValues	Block all setvalues request, this is the only editres request that actually modifies the application, this is in effect stating that the applicaion is read-only.
none	Allow all editres requests.

Remember that these resources are set on any Xaw client, **not editres**. They allow individual clients to keep all or some of the requests editres makes from ever succeeding. Of course, editres is also an Xaw client, so it may also be viewed and modified by editres (rather recursive, I know), these commands can be blocked by setting the **editresBlock** resource on editres itself.

RESOURCES

For **editres** the available application resources are:

numFlashes (Class **NumFlashes)**
: Specifies the number of times the widgets in the client application will be flashed when the **Show Active Widgets** command in invoked.

flashTime (Class **FlashTime)**
: Amount of time between the flashes described above.

flashColor (Class **flashColor)**
: Specifies the color used to flash client widgets. A bright color should be used that will immediately draw your attention to the area being flashed, such as red or yellow.

saveResourcesFile (Class **SaveResourcesFile)**
: This is the file the resource line will be append to when the **Save** button activated in the resource box.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **editres**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Editres  editres
        Paned  paned
                Box  box
                        MenuButton  commands
                                SimpleMenu  menu
                                SmeBSB  sendTree
                                SmeBSB  refreshTree
                                SmeBSB  showClientWidget
                                SmeLine  line
                                SmeBSB  dumpTreeToFile
                                SmeBSB  flashActiveWidgets
                                SmeBSB  getResourceList
                                SmeLine  line
                                SmeBSB  quit
                        MenuButton  treeCommands
                                SimpleMenu  menu
                                SmeBSB  selectAll
                                SmeBSB  unselectAll
                                SmeBSB  invertAll
                                SmeLine  line
                                SmeBSB  selectChildren
                                SmeBSB  selectParent
                                SmeBSB  selectDescendants
                                SmeBSB  selectAncestors
                                SmeLine  line
                                SmeBSB  showWidgetNames
                SmeBSB  showClassNames
                SmeBSB  showWidgetIDs
                SmeBSB  showWidgetWindows
        Paned  hPane
            Panner  panner
            Label  userMessage
            Grip  grip
        Porthole  porthole
            Tree  tree
                Toggle  <name of widget in client>
                .
                .
                .
                TransientShell  resourceBox
                Paned  pane
                Label  resourceLabel
                Form  namesAndClasses
                Toggle  dot
                Toggle  star
                Toggle  any
                Toggle  name
                Toggle  class
                  .
                  .
                  .
                Label  namesLabel
                List  namesList
                Label  constraintLabel
                List  constraintList
                Form  valueForm
                Label  valueLabel
                Text  valueText
                Box  commandBox
                Command  setFile
                Command  save
                Command  apply
                Command  saveAndApply
                Command  cancel
                Grip  grip
        Grip  grip
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/openwin/lib/app-defaults/Editres **-** specifies required resources

SEE ALSO

X11(7), **xrdb**(1), Athena Widget Set

COPYRIGHT

Copyright 1990, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Chris D. Peterson, formerly MIT X Consortium

NAME

fontadmin – font administrator

SYNOPSIS

/usr/openwin/demo/fontadmin

AVAILABILITY

This command is available with the OpenWindows environment. **fontadmin** uses the Motif Graphical User Interface and requires the availability of the Motif toolkit library.

DESCRIPTION

fontadmin is a demonstration program that helps system administrators and everyday users to install, delete and find information about fonts.

USAGE

The main window provides an alphabetized list of the fonts installed on the current host. In addition, font aliases are also indicated with the word "ALIAS" to the right of the name. An alias is not a font, but a logical name which represents a font. Several menu labels and buttons are provided to permit actions on these items. Press MENU to reveal the menu for a label. Press SELECT to activate a button. Menu mnemonics are indicated in square brackets.

Menus:

Fonts > [F]

Delete [D]
: Deletes the currently selected font or fonts. A popup appears to confirm this action.

Install... [I]
: Opens the installation dialog which permits installation of new fonts

Attributes... [A]
: Opens the attributes window which provides more information about the selected font.

Exit [E]
: Quits the font administrator.

Options > [O]

Modify Font Path... [M]
: Opens the Modify Font Path dialog. Important note: this function is restricted for system administrator use only.

Set Sample Text... [S]
: Opens the Set Sample Text dialog which may be used to set the default sample text used by the main window.

Help >

Help...
: Displays on-line man pages in the help dialog.

About Font Administrator...
: Provides information on the title of the application, copyright information and a short description.

Sample Text Point Size >
: Provides the point sizes available for the currently-selected font. This menu allows setting the size of the sample text displayed in the rectangle immediately

below.

Screen Resolution >
: Some bitmap fonts are designed to be used at various screen resolutions. The indicated resolution is in units of dots per inch, horizontal and vertical, respectively. This menu provides the screen resolutions available for the currently-selected font and point size.

Buttons:

Delete
: Deletes the currently selected font or fonts. A popup appears to confirm this action.

Attributes
: Opens the attributes window which provides more information about the selected font.

The Installation Dialog

This window, titled "Font Administrator – Install", permits preview and installation of fonts. The list of fonts which may be installed appears in the main scrolling window.

Explanation of the other features of the Installation Dialog are as follows:

List Fonts From
: Used to choose the source of the fonts to be installed. A source may be Other Media, i.e. CDROM or diskette, or a Directory in the current host's file system. Press SELECT on "Other Media..." to invoke the Other Media dialog or "Directory..." to invoke the file system browser.

Sample Text
: Displays text in the typeface of the currently selected font. The text may be modified by simply typing to the text field. The newly-typed text will remain for the duration of the session. Use the Set Sample Text option on the main window to set this permanently. If multiple fonts are selected, no text appears.

Install to
: Used to set the directory on the current host's filesystem into which the fonts will be installed. The "Directory..." button invokes the file system browser.

Destination
: Indicates the current directory to which fonts will be installed.

Total Disk Space labels
: Total Disk Space Required estimates the amount of disk space required to install the currently selected fonts. Total Disk Space Available indicates the amount of disk space available on the file system to which the Destination directory belongs.

Install
: Used to copy the selected fonts into the current destination directory. In case a font with the same name is already installed on the current host, a confirmation dialog appears. The area immediately below this button shows status information on the results of the installation.

Select All
: Causes all fonts currently in the list to be selected.

Attributes...
: Brings up the font Attributes window already described in conjunction with the main window.

Cancel
: Closes the Installation Dialog.

Help
: Provides more information about the Installation Dialog.

The Other Media Dialog

Used to acquire access to a floppy or CDROM device for **fontadmin.** "Check Fonts From" allows selection of either Floppy (diskette) or CDROM devices.

For systems containing multiple CDROM drives, only the first is available. When a drive type has been selected, clicking SELECT on the "Check" button attempts to access the drive on behalf of **fontadmin.**

If another application has control of the device, a message will appear in the status line at the bottom of the window. In most cases, it will be necessary to go to the other application and release the device from use before **fontadmin** can access it.

Eject
: Used to release **fontadmin**'s lock on the media and eject it from the currently selected drive type.

Close
: Used to take down the Other Media Dialog, but does not release the application's hold on the media. It is necessary to use Eject to accomplish this.

Help
: Provides more information about the Other Media Dialog.

The Attributes Window

This informational window provides secondary data about the currently selected font or alias. If multiple fonts are selected, no information appears. In particular, the following facts are provided:

Font
: The name of the font as it appears in the Main Window or Installation Dialog. If the name selected is an alias, then the Alias field shows the name of the alias and the name of the font that the alias is pointing to.

XLFD Name
: For those familiar with the details of the implementation of the X11 Window System, this field contains the X Logical Font Description name of the font. This is the font name which often appears inside font files. If the name selected is an alias, then this field shows the XLFD name of the font that the alias is pointing to. If a font does not have an XLFD name, the name of the font as it appears in the Main Window or Installation Dialog is displayed.

Location
: The UNIX filesystem directory on the current host in which the font file resides. If the name selected is an alias, then this field is left blank.

Foundry
: The foundry, or font manufacturer or vendor.

Character Set
: The character collection or set supported by this font, generally an industry-standard collection.

Spacing

Indicates whether the font is fixed- or proportionally-spaced. In a fixed-spaced font, all characters are exactly the same width. For example, such fonts are appropriate for tables and for indicating computer text in technical manuals. The majority of fonts have characters whose widths vary depending on the character shapes and are thus labeled Proportional.

Alias
: In case the name selected from the Main Window is an alias, the name of the alias appears here. Otherwise this field and the two following are left blank.

Alias XLFD
: Displays the X Logical Font Description name of the alias. If an alias does not have an XLFD name, the name of the alias as it appears in the Main Window is displayed.

Alias Location
: The UNIX filesystem directory on the current host in which the alias definition resides.

Close
: Takes down the Attributes Window.

Help
: Provides more information about the items in the Attributes Window.

SYSTEM ADMINISTRATOR INFORMATION

Note that the Modify Font Path Dialog window is restricted for system administrator use only.

The Modify Font Path Dialog

This window permits a font administrator to configure the font path of the current host. Because this is a potentially destructive operation, this feature is restricted to the system administrator (the superuser or members of the system administrator group only). This window does not appear unless the invoker has system administrator privileges.

A few words about font path configuration. The font path is akin to the UNIX PATH environment variable. It contains a list of directories which are searched, in order, whenever a font is requested. Unless the font is already cached by the window system, the request is filled from the first directory which has the font name in question, with one exception. The windows server searches all directories for a pre-scaled bitmap font which exactly matches the request before using a scalable font of the same name to satisfy the request. Because it is better to scale from a scalable form of a font than from a bitmap, it is important that directories of scalable fonts, such as **F3, Type1** and **Speedo,** appear earlier in the font path than directories of bitmap fonts having the same font names, such as **75dpi.**

When an ordinary user installs fonts to a directory not part of the persistent font path, this directory is implicitly added to the user's personal font path setting. When fonts are installed to a new directory by the system administrator, no corresponding action is taken. If it is desired that the new directory be added to the system font path, this must be done explicitly.

The window system font path should not be taken lightly. The default font path as it is shipped has been ordered with great care and each element placed in its location for a particular reason. Tampering with this ordering can lead to problems such as strange looking text or even the server failing to restart. Be sure you know what you are doing, make only one change at a time and record the previous state of the font path as the Modify Font Path Dialog has no Undo capability. The window system font path is stored in file *OWconfig* in directory **server/etc** within the OpenWindows installation directory hierarchy. The modifications made to the window system font path will be lost if the new release of OpenWindows is installed.

With the above in mind, this dialog can be used to add directories of newly-installed fonts to the existing font path, to delete directories which are no longer being used or insert new font directories earlier in the font path.

Modify Font Path for
: Used to choose between potential multiple copies of OpenWindows on the current host. The default choice is the OpenWindows installation in **/usr/openwin.** If however other installations exist, as might be the case on an NFS server serving heterogeneous architectures for example, it may be specified by selecting the directory at the top of the installation via the file system browser available from the "Directory..." button.

Current Font Path Listing for
: Indicates which copy of OpenWindows is currently being examined. The attached scrolling list shows, in order, the directories which comprise the font path.

Move Up
: Pressing SELECT on this button exchanges the currently selected font path element with the one immediately preceding it, unless the selected element is first, in which case, it does nothing.

Move Down
: Exchanges the currently selected font path element with the one immediately following it, unless the selected element is last in which case it does nothing.

Delete
: Deletes the selected font path element.

Add New
: Adds the font path element which is currently listed under New Font Path Element into the current font path. The new element is always added to the end of the list.

New Font Path Element
: This field contains the directory which was the most recent destination for installed fonts in the current **fontadmin** session. If no installation has occurred in the current session, it is left blank. This file system browser may be invoked with the "Directory..." button to choose another directory to add to the font path and it may be added via the "Add New" button.

OK
: Used to apply the changes made and close the dialog. Check the status line of the main window for messages.

Apply
: Applies the changes made without closing the dialog. Check the status line of this dialog for messages.

Reset	Backtracks to the state of the dialog just after the last Apply, or if Apply has not been used, to the state present at the invocation of the dialog.
Cancel	Ignores all changes made since the last Apply and closes the dialog.
Help	Provides more information about the features of the Modify Font Path dialog.

The Set Sample Text Dialog

This dialog causes **fontadmin** to remember the text used in the Sample Text fields of the Main Window and Installation Dialog between sessions.

Sample Text to Display	The field immediately below this label is used to type in the new sample text.
OK	Applies the changes and closes the dialog.
Apply	Applies the changes without closing the dialog.
Reset	Returns the dialog to the state just after the last Apply in the current session of the dialog or, if Apply has not been used in the current session, to the state of the dialog at the start of the current invocation.
Cancel	Backtracks all changes to the last Apply in the current session, or if Apply has not been used, to the state at the start of the current session and closes the dialog.
Help	Provides more information about the features of Set Sample Text.

SUPPORTED FORMATS

fontadmin supports the following font formats:

Adobe Type 1 – ASCII (.pfa)
Adobe Type 1 – Binary (.pfb)
Adobe Type 0
Adobe Type 3
PCF bitmap (.pcf)

Fonts of other formats can still be used by the system, but are not visible to **fontadmin.**

FILES

/usr/openwin/etc/owfontpath – OpenWindows font path setting

$HOME/.OWfontpath – personal font path setting

$HOME/.fontadmrc – personal tool property settings

SEE ALSO

font_utils(6), **fp_utils**(6), **xlsfonts**(1), **mkfontdir**(1), **makepsres**(1), **makebdf**(1)

NAME

fontadd – install fonts in a directory

fontls – list fonts in a directory

fontrm – remove installed fonts from a directory

SYNOPSIS

/usr/openwin/demo/fontadd [**–e** *dir*] [**–v**] [**–f**] [*font_name* . . .] *fromdir todir*

/usr/openwin/demo/fontls [*–x*] [*dir*]

/usr/openwin/demo/fontrm [**–e** *dir*] [**–v**] [**–f**] [*font_name* . . .] *dir*

DESCRIPTION

These utilities permit manipulation of font files by font name (which is a string stored inside each font file) rather than by the more clumsy approach of direct access to the UNIX files.

These commands are available with the OpenWindows environment.

fontadd

fontadd copies fonts specified by *font_name* from the directory *fromdir* to the directory *todir*. If no font is specified on the command line, it is assumed that fonts will be specified from the standard input. Since *font_names* contain square brackets and blank spaces, they should be quoted when specified on the command line.

If the user is the superuser, the font path of OpenWindows installed in **/usr/openwin** is updated if necessary by appending *todir* to the path.

The command asks for confirmation when a conflict arises. A conflict arises if a font to be added is already in the system. If the font is already in the system and is read-only, that font will not be installed. The user will be prompted to continue with the installation of the next font in the list. If the font is not read-only, the user is presented with a choice of overwriting the installed font with the new font.

Newly-installed fonts will not be available until the next invocation of the desktop.

fontls

fontls lists fonts in a directory. The font names are displayed on the standard output. Each font name is separated by the newline character. The output can be redirected to a file to be used later as an input to **fontrm** and **fontadd** commands. *dir* specifies a directory to list fonts from. If no directory is specified, the current directory is assumed.

fontrm

fontrm removes fonts from the directory *dir*. If no font is specified on the command line, it is assumed that fonts will be specified from the standard input. Since *font_names* contain square brackets and blank spaces, they should be quoted when specified on the command line.

If the user is the superuser, the font path of OpenWindows installed in **/usr/openwin** is updated if necessary, that is, if the directory *dir* is empty after deleting fonts, it is removed from the font path.

The command asks for confirmation before deleting each font and when a conflict arises. A conflict arises when an attempt to remove a read-only font or a system font is made.

OPTIONS

–*x* List fonts by **XLFD** name. This option is available only for the **fontls** command. Bitmap (PCF) and Type 1 fonts in the directory will be listed by their **XLFD** names. Font types that have no **XLFD** name and are accessible through Display PostScript only (such as Type 3 and Type 0) are not listed.

–**e** *dir* Meaningful only if the user is the superuser. The font path of the copy of OpenWindows in directory *dir* is updated if necessary, that is, if as a result of removing fonts, the directory becomes empty, it is removed from the font path. In addition, if fonts are installed in a directory not in the font path, the font path is updated by appending the directory *todir* to the font path of OpenWindows installation in directory *dir*. This option is not available with the **fontls** command.

-v Verbose mode. Each font that is removed or installed is displayed on the standard output. This option is not available with the **fontls** command.

-f If this option is specified, no user interaction takes place. If conflicting font is read-only, it is not deleted or the new font is not installed. Otherwise, the conflicting font is deleted or it is replaced by the new font. This option is not available with the **fontls** command.

FONT NAME INFORMATION

A font name represents either multiple X bitmap fonts or one X outline font or one PostScript font. Multiple X bitmap fonts exist in the system with the same characteristics such as font foundry, family, typeface etc. but with different point sizes and screen resolution. A single font name groups all such X bitmap fonts together. In contrast, the same outline and PostScript font is scaled for different point sizes and screen resolution. Therefore, only one X outline or PostScript font exists in the system with the same characteristics.

A *font_name* is constructed by the following rules:

For a font with **XLFD** name, the font name is constructed by concatenating the following fields from **XLFD** name in the order they appear: *foundry* , *family* , *setwidth* , *add style* , *weight* , *slant* , *character set* , *format* and enclosing them in square brackets i.e. '[' and ']'.

The following additional rules apply:

1. Capitalize the first letter of each field's string in the font name.
2. Convert the *slant* field from a field in **XLFD** name to a field in font name as follows:

XLFD	FONT NAME
"R"	"" (the empty string)
"I"	"Italic"
"O"	"Oblique"
"RI"	"Reverse Italic"
"RO"	"Reverse Oblique"
"OT"	"" (the empty string)

3. Replace "Normal" with the empty string when it appears in the *setwidth* field.
4. Replace "Medium" with the empty string when it appears in the *Weight* field.
5. Convert fields *character set registry* and *character set encoding* from **XLFD** name to font

name fields as follows:

XLFD	**FONT NAME**
"DEC.CNS11643.1986-2"	"DEC-Chinese-11643"
"DEC.DTSCS.1990-2"	"DEC-Taiwan-Supplemental"
"GB2312.1980-0"	"PRC-Hanzi-GL"
"GB2312.1980-1"	"PRC-Hanzi-GR"
"ISO646.1991-IRV"	"ISO-646"
"ISO8859-1"	"ISO-Latin-1"
"ISO8859-2"	"ISO-Latin-2"
"ISO8859-3"	"ISO-Latin-3"
"ISO8859-4"	"ISO-Latin-4"
"ISO8859-5"	"ISO-Latin / Cyrillic"
"ISO8859-6"	"ISO-Latin / Arabic"
"ISO8859-7"	"ISO-Latin / Greek"
"ISO8859-8"	"ISO-Latin / Hebrew"
"ISO8859-9"	"ISO-Latin-5"
"JISX0201.1976-0"	"Roman-Katakana"
"JISX0208.1983-0"	"Kanji-1983"
"JISX0208.1983-1"	"Kanji-1983-1"
"JISX0208.1990-0"	"Kanji-1990"
"JISX0212.1990-0"	"Kanji-Supplement-1990"
"KSC5601.1987-0"	"Korean-1987"
"KSC5601.1987-1"	"Korean-1987-1"

If the **XLFD** *registry* and *encoding* fields do not have an entry in the above table, the *character set* string is constructed by concatenating *registry* and *encoding* fields with a "-" character separating them.

For a font with only a **PostScript** name, the *font_name* is constructed by concatenating the font's **PostScript** name with the font type, separating them by a blank character and enclosing them in square brackets.

EXAMPLES

The following bitmap fonts with the **XLFD** names

"-monotype-gill sans-bold-i-normal--10-100-72-72-p-50-iso8859-1"

"-monotype-gill sans-bold-i-normal--12-120-72-72-p-60-iso8859-1"

"-monotype-gill sans-bold-i-normal--14-140-72-72-p-69-iso8859-1"

are referred to by the font name

[Monotype Gill sans Bold Italic Latin-1 Bitmap]

A Type 3 font with the **PostScript** name "Hershey-Greek" is referred to by the font name

[Hershey-Greek Type-3]

FILES

/usr/openwin/server/etc/OWconfig – OpenWindows font path setting

$HOME/.OWfontpath – personal font path setting
$HOME/.fontadmrc – personal tool property settings

SEE ALSO

fontadmin(6), **fp_utils**(6), **xlsfonts**(1)

NAME

fpadd – add a font path element

fpls – list font path elements in a comma separated list

fprm – remove a font path element

SYNOPSIS

/usr/openwin/demo/fpadd [**–e** *dir*] *position font_path_element*

/usr/openwin/demo/fpls [**–e** *dir*]

/usr/openwin/demo/fprm [**–e** *dir*] *font_path_element*

DESCRIPTION

A running Solaris desktop has an associated *font path* which is simply a list of directories which are searched sequentially for fonts (much as the shell uses the PATH environment variable to find commands). These commands are to be used to view the current font path and to modify the font path which will be used on future invocations of the desktop.

These commands are available with and affect the OpenWindows environment.

fpadd

Adds the specified element to the font path. This command is available only to the superuser. The specified element is added to the font path of the copy of OpenWindows installed in **/usr/openwin** by default. The values for *position* are "first", "last" or a positive integer, indicating into which position in the font path the new element should be inserted.

fpls

Lists elements in a font path. Each element is separated by the comma character. If the user is the superuser, the font path of the OpenWindows installation in **/usr/openwin** is listed by default.

fprm

Removes an element from the font path. This command is available only to the superuser. The specified element is removed from the font path of the OpenWindows installed in **/usr/openwin** by default.

OPTIONS

–e *dir* Meaningful only if the user is the superuser. The font path of the OpenWindows installation in directory *dir* is listed or modified depending on the command.

FILES

/usr/openwin/server/etc/OWconfig – OpenWindows font path setting

$HOME/.OWfontpath – personal font path setting

SEE ALSO

fontadmin(6), **font_utils**(6), **xset(1)**

NAME

fs – X font server

SYNOPSIS

fs [**–config** *configuration_file*] [**–ls** *listen-socket*] [**–port** *tcp_port*]

AVAILABILITY

SUNWxwfs

DESCRIPTION

fs is the X Window System font server. It supplies fonts to X Window System display servers.

fs can be started manually or automatically whenever a client application requests its service at port 7100. Automatic starting can be enabled or disabled through the **fsadmin**(1) command.

OPTIONS

–config *configuration_file*
: Specifies the configuration file the font server will use. The default configuration file is **/usr/openwin/lib/X11/fontserver.cfg**

–ls *listen-socket*
: Specifies a file descriptor which is already set up to be used as the listen socket. This option is only intended to be used by the font server itself when automatically spawning another copy of itself to handle additional connections.

–port *tcp_port*
: Specifies the TCP port number on which the server will listen for connections.

SIGNALS

SIGTERM
: This causes the font server to exit cleanly.

SIGUSR1
: This signal is used to cause the server to re-read its configuration file.

SIGUSR2
: This signal is used to cause the server to flush any cached data it may have.

SIGHUP This signal is used to cause the server to reset, closing all active connections and re-reading the configuration file.

CONFIGURATION

The configuration language is a list of keyword and value pairs. Each keyword is followed by an '=' and then the desired value.

Recognized keywords include:

catalogue (list of string)
: Ordered list of font path element names. Use of the keyword "catalogue" is very misleading at present, the current implementation only supports a single catalogue ("all"), containing all of the specified fonts.

alternate-servers (list of string)
: List of alternate servers for this font server.

client-limit (cardinal)
: Number of clients this font server will support before refusing service. This is

useful for tuning the load on each individual font server.

clone-self (boolean)
: Whether this font server should attempt to clone itself when it reachs the client-limit.

default-point-size (cardinal)
: The default pointsize (in decipoints) for fonts that don't specify.

default-resolutions (list of resolutions)
: Resolutions the server supports by default. This information may be used as a hint for pre-rendering, and substituted for scaled fonts which do not specify a resolution.

error-file (string)
: Filename of the error file. All warnings and errors will be logged here.

port (cardinal)
: TCP port on which the server will listen for connections.

use-syslog (boolean)
: Whether **syslog**(3) (on supported systems) is to be used for errors.

EXAMPLE

```
#
# sample font server configuration file
#

# allow a max of 10 clients to connect to this font server
client-limit = 10

# when a font server reaches its limit, start up a new one
clone-self = on

# alternate font servers for clients to use
alternate-servers = hansen:7001,hansen:7002

# where to look for fonts
# the first is a set of Speedo outlines, the second is a set of
# misc bitmaps and the last is a set of 100dpi bitmaps
#
catalogue = /usr/openwin/lib/X11/fonts/Type1,
        /usr/openwin/lib/X11/fonts/Speedo,
        /usr/openwin/lib/X11/fonts/misc,
        /usr/openwin/lib/X11/fonts/75dpi,
        /usr/openwin/lib/X11/fonts/100dpi,
        /usr/openwin/lib/X11/fonts/Xt+

# in 12 points, decipoints
default-point-size = 120
```

```
# 100 x 100 and 75 x 75
default-resolutions = 100,100,75,75
```

FONT SERVER NAMES

One of the following forms can be used to name a font server that accepts TCP connections:

tcp / *hostname*:*port*
tcp / *hostname*:*port* / *cataloguelist*

The *hostname* specifies the name (or decimal numeric address) of the machine on which the font server is running. The *port* is the decimal TCP port on which the font server is listening for connections. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *tcp/expo.lcs.mit.edu:7000* , *tcp/18.30.0.212:7001/all*

One of the following forms can be used to name a font server that accepts DECnet connections:

decnet / *nodename*::font$*objname*
decnet / *nodename*::font$*objname* / *cataloguelist*

The *nodename* specifies the name (or decimal numeric address) of the machine on which the font server is running. The *objname* is a normal, case-insensitive DECnet object name. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *DECnet/SRVNOD::FONT$DEFAULT* , *decnet/44.70::font$special/symbols*

SEE ALSO

fsadmin(1), **fsinfo**(1), **fslsfonts**(1), **fstobdf**(1), **showfont**(1), **X11**(7), *Font server implementation overview*

BUGS

Multiple catalogues should be supported.

COPYRIGHT

Copyright 1991, Network Computing Devices, Inc Copyright 1991, Massachusetts Institute of Technology
See **X11**(7) for a full statement of rights and permissions.

AUTHORS

Dave Lemke, Network Computing Devices, Inc
Keith Packard, Massachusetts Institute of Technology

NAME

fsadmin – font server administration utility

SYNOPSIS

fsadmin [**–e** | **–d** | **–usage**]

AVAILABILITY

SUNWxwfs

DESCRIPTION

fsadmin is a font server administration utility. **fsadmin** allows users to control whether or not the font server is started automatically by **inetd**(1M). If no options are present **fsadmin** will display the current status of the font server (either enabled or disabled).

OPTIONS

–e	enable: configure **inetd** to start the font server automatically
–d	disable: configure **inetd** to not start the font server automatically
–usage	display list of options for **fsadmin**

SEE ALSO

fs(1), **inetd**(1M), **X11**(7)

COPYRIGHT

See **X11**(7) for a full statement of rights and permissions.

NAME

fsinfo – font server information utility

SYNOPSIS

fsinfo [**–s** *host:port*]

DESCRIPTION

fsinfo is a utility for displaying information about an X font server. It is used to examine the capabilities of a server, the predefined values for various parameters used in communicating between clients and the server, and the font catalogues and alternate servers that are available.

OPTIONS

–s *host:port* This option specifies the X font server to contact.

EXAMPLE

The following shows a sample produced by **fsinfo.**

```
name of server:  hansen:7000
version number:           1
vendor string:    Font Server Prototype
vendor release number: 17
maximum request size:  16384 longwords (65536 bytes)
number of catalogues:    1
         all
Number of alternate servers: 2
   #0      hansen:7001
   #1      hansen:7002
number of extensions:    0
```

ENVIRONMENT

FONTSERVER
To get the default fontserver.

SEE ALSO

fs(1), **fslsfonts**(1)

COPYRIGHT

Copyright 1991, Network Computing Devices, Inc
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Dave Lemke, Network Computing Devices, Inc

NAME

fslsfonts – server font list displayer for X font server

SYNOPSIS

fslsfonts [–**1**] [– **C**] [–**fn** *pattern*] [–**l** | –**ll** | –**lll**] [–**m**] [–**n** *columns*] [–**server** *host:port*] [– **u**] [–**w** *width*]

DESCRIPTION

fslsfonts lists the fonts that match the given *pattern*. The wildcard character "*" may be used to match any sequence of characters (including none), and "?" to match any single character. If no *pattern* is given, "*" is assumed.

The "*" and "?" characters must be quoted to prevent them from being expanded by the shell.

OPTIONS

–**server** *host:port*
This option specifies the X font server to contact.

–l, –ll, –lll
These options indicates that medium, long, and very long listings, respectively, should be generated for each font.

–**m** This option indicates that long listings should also print the minimum and maximum bounds of each font.

–**C** This option indicates that listings should use multiple columns. This is the same as **-n 0**

–**1** This option indicates that listings should use a single column. This is the same as **-n 1.**

–**w** *width*
This option specifies the width in characters that should be used in figuring out how many columns to print. The default is 79.

–**n** *columns*
This option specifies the number of columns to use in displaying the output. By default, it will attempt to fit as many columns of font names into the number of character specified by **-w** *width.*

–**u** This option indicates that the output should be left unsorted.

SEE ALSO

fs(1), **showfont**(1), **xlsfonts**(1)

ENVIRONMENT

FONTSERVER
to get the default host and port to use.

BUGS

Doing ‘‘fslsfonts -l’’ can tie up your server for a very long time. This is really a bug with single-threaded non-preemptable servers, not with this program.

COPYRIGHT

AUTHOR

Dave Lemke, Network Computing Devices, Inc

NAME

fstobdf – BDF font generator

SYNOPSIS

fstobdf –**fn** *fontname* [–**s** *host:port*]

DESCRIPTION

The **fstobdf** program reads a font from a font server and prints a BDF file on the standard output that may be used to recreate the font. This is useful in testing servers, debugging font metrics, and reproducing lost BDF files.

OPTIONS

–**fn** *fontname*
This option specifies the font for which a BDF file should be generated.

–**s** *host:port*
This option specifies the server from which the font should be read.

SEE ALSO

fs(1), **bdftosnf**(1), **fslsfonts**(1)

COPYRIGHT

Copyright 1990, Network Computing Devices
Copyright 1990, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Olaf Brandt, Network Computing Devices
Dave Lemke, Network Computing Devices
Jim Fulton, MIT X Consortium

NAME

ico – animate an icosahedron

SYNOPSIS

ico [*display list*] [**-geometry** *geom*] [**-d** *pattern*] [**-i**]

DESCRIPTION

Ico displays a wire-frame rotating polyhedron, with hidden lines removed, or a solid-fill polyhedron with hidden faces removed. There are a number of different polyhedra available; adding a new polyhedron to the program is quite simple.

OPTIONS

-geometry *geom*	Specify the size and / or location of the window.
-d *pattern*	Specify a bit pattern for drawing dashed lines for wire frames.
-i	Use inverted colors for wire frames.

For each display specified, **ico** creates another window. The total geometry is split vertically amongst these windows, and the ico bounces between them. The end result is that **ico foo:0 bar:0** would result in ico bouncing half on display **foo:0** and half on **bar:0.**

SEE ALSO

X11(7), **ico2**(6)

BUGS

Doesn't deal too well with being resized.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

NAME

ico2 – animate an icosahedron or other polyhedron

SYNOPSIS

ico2 [**-r**] [**-d** *pattern*] [**-i] [-dbl**] [**-faces**] [**-noedges**] [**-sleep** *n*] [**-obj** *object*] [**-objhelp**] [**-colors** *color-list*]

DESCRIPTION

Ico2 displays a wire-frame rotating polyhedron, with hidden lines removed, or a solid-fill polyhedron with hidden faces removed. There are a number of different polyhedra available; adding a new polyhedron to the program is quite simple.

OPTIONS

-r Display on the root window instead of creating a new window.

-d *pattern*
Specify a bit pattern for drawing dashed lines for wire frames.

-i Use inverted colors for wire frames.

-dbl Use double buffering on the display. This works for either wire frame or solid fill drawings. For solid fill drawings, using this switch results in substantially smoother movement. Note that this requires twice as many bit planes as without double buffering. Since some colors are typically allocated by other programs, most eight-bit-plane displays will probably be limited to eight colors when using double buffering.

-faces Draw filled faces instead of wire frames.

-noedges
Don't draw the wire frames. Typically used only when -faces is used.

-sleep *n*
Sleep n seconds between each move of the object.

-obj *object*
Specify what object to draw. If no object is specified, an icosahedron is drawn.

-objhelp
Print out a list of the available objects, along with information about each object.

-colors *colorlist*
Specify what colors should be used to draw the filled faces of the object. If less colors than faces are given, the colors are reused.

ADDING POLYHEDRA

If you have the source to ico, it is very easy to add more polyhedra. Each polyhedron is defined in an include file by the name of obj*name*.h, where *name* is something related to the name of the polyhedron. The format of the include file is defined in the file polyinfo.h. Look at the file objcube.h to see what the exact format of an obj*name*.h file should be, then create your obj*name*.h file in that format.

After making the new obj*name*.h file (or copying in a new one from elsewhere), simply do a 'make depend'. This will recreate the file allobjs.h, which lists all of the obj*name*.h files. Doing a 'make' after this will rebuild ico with the new object information.

BUGS

If *-dbl* or *-colors* is specified on a monochrome screen, or if the number of bit planes needed is more than what is available, the server hangs (this is on a Sun 3 / 60, patches up to 80 applied).

A separate color cell is allocated for each name in the *-colors* list, even when the same name may be specified twice.

SEE ALSO

ico(6)

NAME

imake – C preprocessor interface to the make utility

SYNOPSIS

imake [**-D***define*] [**-I***dir*] [**-T***template*] [**-f** *filename*] [**-s** *filename*] [**-e**] [**-v**]

DESCRIPTION

imake is used to generate **Makefiles** from a template, a set of **cpp**(1) macro functions, and a per-directory input file called an **Imakefile**. This allows machine dependencies (such has compiler options, alternate command names, and special **make**(1S) rules) to be kept separate from the descriptions of the various items to be built.

OPTIONS

The following command line options may be passed to **imake**:

–**D***define*
: This option is passed directly to **cpp**. It is typically used to set directory-specific variables. For example, the X Window System uses this flag to set *TOPDIR* to the name of the directory containing the top of the core distribution and *CURDIR* to the name of the current directory, relative to the top.

–**I***directory*
: This option is passed directly to **cpp**(1). It is typically used to indicate the directory in which the **imake** template and configuration files may be found.

–**T***template*
: This option specifies the name of the master template file (which is usually located in the directory specified with *–I*) used by **cpp**. The default is **Imake.tmpl**.

–**f** *filename*
: This option specifies the name of the per-directory input file. The default is **Imakefile**.

–**s** *filename*
: This option specifies the name of the **make** description file to be generated but **make** should not be invoked. If the *filename* is a dash (-), the output is written to **stdout**. The default is to generate, but not execute, a **Makefile**.

–**e**
: This option indicates the **imake** should execute the generated **Makefile**. The default is to leave this to the user.

–**v**
: This option indicates that **imake** should print the **cpp** command line that it is using to generate the **Makefile**.

HOW IT WORKS

imake invokes **cpp** with any –**I** or **-D** flags passed on the command line and passes it the following 3 lines:

```
#define IMAKE_TEMPLATE "Imake.tmpl"
#define INCLUDE_IMAKEFILE "Imakefile"
#include IMAKE_TEMPLATE
```

where *Imake.tmpl* and *Imakefile* may be overridden by the –**T** and –**f** command options, respectively. If the **Imakefile** contains any lines beginning with a '#' character that is not followed by a **cpp** directive (**#include**, **#define**, **#undef**, **#ifdef**, **#else**, **#endif**, or **#if**), **imake** will make a temporary **makefile** in which the '#' lines are prepended with the string ‘‘/**/’’ (so that **cpp** will copy the line into the **Makefile** as a comment).

The **Imakefile** reads in file containing machine-dependent parameters (specified as **cpp** symbols), a site-specific parameters file, a file containing **cpp** macro functions for generating **make** rules, and finally the **Imakefile** (specified by INCLUDE_IMAKEFILE) in the current directory. The **Imakefile** uses the macro functions to indicate what targets should be built; **imake** takes care of generating the appropriate rules.

The rules file (usually named **Imake.rules** in the configuration directory) contains a variety of **cpp** macro functions that are configured according to the current platform. **imake** replaces any occurrences of the string ‘‘@@’’ with a newline to allow macros that generate more than one line of **make** rules. For example, the macro

```
#define     program_target(program, objlist)                    @@\
program:    objlist                                             @@\
            $(CC) -o $@ objlist $(LDFLAGS)
```

when called with *program_target(foo, foo1.o foo2.o)* will expand to

```
foo:        foo1.o foo2.o
            $(CC) -o $@ foo1.o foo2.o $(LDFLAGS)
```

On systems whose **cpp** reduces multiple tabs and spaces to a single space, **imake** attempts to put back any necessary tabs (**make** is very picky about the difference between tabs and spaces). For this reason, colons (:) in command lines must be preceded by a backslash (\).

USE WITH THE X WINDOW SYSTEM

The X Window System uses **imake** extensively, for both full builds within the source tree and external software. As mentioned above, two special variables, *TOPDIR* and *CURDIR* set to make referencing files using relative path names easier. For example, the following command is generated automatically to build the **Makefile** in the directory **lib/X/** (relative to the top of the sources):

```
% ../.././config/imake -I../.././config \
            -DTOPDIR=../../. -DCURDIR=./lib/X
```

When building X programs outside the source tree, a special symbol **UseInstalled** is defined and *TOPDIR* and *CURDIR* are omitted. If the configuration files have been properly installed, the script **xmkmf**(1) may be used to specify the proper options:

```
% xmkmf
```

The command **make Makefiles** can then be used to generate **Makefiles** in any subdirectories.

FILES

usr/tmp/tmp-imake.*nnnnnn*	temporary input file for **cpp**
usr/tmp/tmp-make.*nnnnnn*	temporary input file for **make**
usr/ccs/lib/cpp	default C preprocessor

SEE ALSO

make(1S), **execvp**(2), **xmkmf**(1)
S. I. Feldman *Make – A Program for Maintaining Computer Programs*

ENVIRONMENT VARIABLES

The following environment variables may be set, however their use is not recommended as they introduce dependencies that are not readily apparent when **imake** is run:

IMAKEINCLUDE
If defined, this should be a valid include argument for the C preprocessor. E.g. *-I* **/usr/include/local**. Actually, any valid **cpp** argument will work here.

IMAKECPP
If defined, this should be a valid path to a preprocessor program. E.g. **/usr/local/cpp**. By default, **imake** will use **/lib/cpp**.

IMAKEMAKE
If defined, this should be a valid path to a make program. E.g. **/usr/local/make**. By default, **imake** will use whatever **make** program is found using **execvp**(2).

BUGS

Comments should be preceded by ‘‘/**/#’’ to protect them from **cpp**.

AUTHOR

Todd Brunhoff, Tektronix and MIT Project Athena; Jim Fulton, MIT X Consortium

NAME

kbd_mode – change the keyboard translation mode

SYNOPSIS

kbd_mode -a | -n | -e | -u

DESCRIPTION

kbd_mode sets the translation mode of the console's keyboard (**/dev/kbd**) to one of the four values defined for **KIOCTRANS** in **kb**(7)**.** This is useful when a program that resets the translation mode terminates abnormally and fails to restore the original translation mode.

Note that SunView desires translated events (**kbd_mode -e**), while **Xsun**(1) desires untranslated events (**kbd_mode -u**). See below for an explanation of the **-e** and **-u** options.

OPTIONS

-a ASCII: the keyboard will generate simple ASCII characters.

-n None: the keyboard will generate unencoded bytes – a distinct value for up and down on each switch on the keyboard.

-e Events: the keyboard will generate SunWindows input events with ASCII characters in the *value* field.

-u Unencoded: the keyboard will generate input events with unencoded bytes in the *value* field such as those desired by the **Xsun**(1) server.

FILES

/dev/kbd
/usr/openwin/bin/kbd_mode

SEE ALSO

kb(7)

NAME

keytablemap – maps keyboard type and layout to keytable

SYNOPSIS

/usr/openwin/etc/keytables/keytable.map

DESCRIPTION

The **keytable map** contains information that maps keyboard type and layout to the appropriate keytable file. The keyboard **type** and **layout** are generally obtained from the kernel using appropriate ioctl calls. The keyboard layout is assumed to be US layout (code=0) on Sun keyboards prior to type 4. The layout is determined by a keyboard DIP switch setting on type-4 and subsequent types. The keytable file contains a mapping of keycap symbols to phyical keys.

For each keyboard type and layout, a single line should be present with the following information:

keyboardtype   keyboardlayout   keytablefilename

Any characters following a white space after the keytablefilename field, through the end of line are disregarded. The fields are separated by any number of blanks and / or TAB characters. A leading '**#**' indicates the beginning of a comment, which may appear only as the first character on the line. Such comment lines are ignored by routines that read the file.

EXAMPLE

An example of a keytablemap file appears below.

```
{
# keytable.map
#
# The keytable map associates a keyboard with its
# corresponding keytable.  Keytables are located in
# /usr/openwin/etc/keytables.
#
# The keyboard type and layout are generally obtained
# from the kernel using appropriate ioctl calls.  The
# keyboard type is stored in a keyboard ROM.  The
# keyboard layout is assumed to be 0 on Sun keyboards
# prior to type 4 and is determined by a DIP switch
# setting on type-4 and later keyboards.
#
# Notes:
# To test a new keytable before installing it, copy it
# to $HOME/.keytable, start the server and test.
# (Remember to remove $HOME/.keytable after testing.)
#
# Format of an entry:
# keyboard_type  keyboard_layout  keytable_filename
#
```

```
#Type   Layout  Filename
2       0       US2.kt
3       0       US3.kt
4       0       US4.kt        # Sun US Type 4 keyboard
4       2       FranceBelg4.kt
4       3       Canada4.kt
# -- Truncated --
```

FILES **/usr/openwin/etc/keytables/keytable.map**

NOTES This file is intended to be used only by the OpenWindows server.

NAME

listres – list resources in widgets

SYNOPSIS

listres [**-all**] [**-nosuper**] [**-variable**] [**-top** *name*] [**-format** *printf-string*]

DESCRIPTION

The **listres** program generates a list of a widget's resource database. The class in which each resource is first defined, the instance and class name, and the type of each resource is listed. If no specific widgets or the **-all** switch are given, a two-column list of widget names and their class hierarchies is printed.

OPTIONS

Listres accepts all of the standard toolkit command line options along with those listed below:

-all
This option indicates that **listres** should print information for all known widgets and objects.

-nosuper
This option indicates that resources that are inherited from a superclass should not be listed. This is useful for determining which resources are new to a sub-class.

-variable
This option indicates that widgets should be identified by the names of the class record variables rather than the class name given in the variable. This is useful for distinguishing subclasses that have the same class name as their superc-lasses.

-top *name*
This option specifies the name of the widget to be treated as the top of the hierarchy. Case is not significant, and the name may match either the class vari-able name or the class name. The default is ‘‘core’’.

-format *printf-string*
This option specifies the *printf*-style format string to be used to print out the name, instance, class, and type of each resource.

SEE ALSO

X11(7), **xrdb**(1), appropriate widget documents

BUGS

On operating systems that do not support dynamic linking of run-time routines, this pro-gram must have all of its known widgets compiled in. The sources provide several tools for automating this process for various widget sets.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

makebdf – create bitmap files from scalable F3 or X11/NeWS font files

SYNOPSIS

makebdf [–**a**] [–**A**] [–**e** *filename*] [–**f** *n*] [–**m** | –**M**] [–**p** | –**P**] [–**s**] [–**v** | –**V**] [–*values*] [–**x**]

DESCRIPTION

makebdf takes a list of Scalable F3 Font Format outlines (**.f3b** files) or X11/NeWS Bitmap fonts (**.fb** files) and generates ASCII bitmap fonts in either Bitmap Distribution Format (**.bdf**) or the Adobe Font Bitmap (**.afb**) format and Adobe Font Metrics (**.afm**) files.

makebdf will not create bitmaps if screen protection for the F3 font has been activated or if any form of protection exists and the requested size is 24 or higher.

NOTES

Options with ∗ apply to F3 files only, they are ignored for ∗.fb files.

OPTIONS

-a Produce font metric (**.afm**) files only.∗

-A Produce Adobe Font Bitmap (.afb) format files.

-e *filename* Specify an encoding map file to be loaded along with the font. This option is only required if the encoding specified by the font is other than ISO Latin-1, Symbol or Dingbats.

-f*n* Force the length of the base part of the output filename to be at most *n* characters. The default is 32.

-m Enable generation of **.afm** files (the default).∗

-M Disable generation of **.afm** files.∗

-p Preserve existing files. If **-p** is selected, then just before **makebdf** writes a file it will check to see if it already exists. If it does, the file will be skipped. This is useful in situations where you have some handbuilt *.afb* and *.afm* files, and just want to fill in the missing ones.

-P Don´t preserve existing files (the default).

-s Produce a synthetic encoding map file in <*encodingname*>.**map**.∗

-v Verbose: print messages indicating what´s going on (the default).

-V Work silently.

-sizes A comma separated list of pixel sizes for which **.afb** files should be generated. The default is *6,8,10,12,14,16,18* .*

-x Produce bitmap fonts in BDF2.1 format. Bitmap font file suffix will be **.bdf**. This is the default.

EXAMPLES

```
example% makebdf *.f3b

example% makebdf -e latin.map *.fb

example% makebdf -p -4,5,6,7,8,9,10,11,12,14,16,18,20,24 *.f3b
```

NAME

makedepend – create dependencies in makefiles

SYNOPSIS

makedepend [–**D** *name=def*] [–**D** *name*] [–**I** *includedir*] [–**f** *makefile*] [–**o** *objsuffix*]
[–**s** *string*] [–**w** *width*] [– – *otheroptions*] *sourcefile*

DESCRIPTION

Makedepend reads each *sourcefile* in sequence and parses it like a C-preprocessor, processing all *#include, #define, #undef, #ifdef, #ifndef, #endif, #if* and *#else* directives so that it can correctly tell which *#include,* directives would be used in a compilation. Any *#include* directives can reference files having other *#include* directives--parsing will occur in these files as well.

Every file that a *sourcefile* includes, directly or indirectly, is what **makedepend** calls a *dependency*. These dependencies are then written to a **makefile** in such a way that **make**(1S) will know which object files must be recompiled when a dependency has changed.

By default, **makedepend** places its output in the file named **makefile** if it exists, otherwise **Makefile** An alternate makefile may be specified with the **-f** option. It first searches the makefile for the line

# DO NOT DELETE THIS LINE -- make depend depends on it.

or one provided with the **-s** option, as a delimiter for the dependency output. If it finds it, it will delete everything following this to the end of the **makefile** and put the output after this line. If it doesn't find it, the program will append the string to the end of the **makefile** and place the output following that. For each *sourcefile* appearing on the command line, **makedepend** puts lines in the **makefile** of the form

sourcefile.**o:** *dfile* ...

Where **sourcefile.o** is the name from the command line with its suffix replaced with **.o**, and *dfile* is a dependency discovered in a *#include* directive while parsing *sourcefile* or one of the files it included.

EXAMPLE

Normally, **makedepend** will be used in a **makefile** target so that typing "make depend" will bring the dependencies up to date for the **makefile**. For example,

```
SRCS = file1.c file2.c ...
CFLAGS = -O -DHACK -I../foobar -xyz
depend:
        makedepend -- $(CFLAGS) -- $(SRCS)
```

OPTIONS

Makedepend will ignore any option that it does not understand so that you may use the same arguments that you would for **cc**(1B).

–**D** *name=def* or –**D** *name* Define. This places a definition for *name* in **makedepend's** symbol table. Without =*def* the symbol becomes defined as "1".

–**I** *includedir* Include directory. This option tells **makedepend** to prepend *includedir* to its list of directories to search when it encounters a *#include* directive. By default, **makedepend** only searches **/usr/include**

–f *makefile* Filename. This allows you to specify an alternate **makefile** in which **makedepend** can place its output.

–o *objsuffix* Object file suffix. Some systems may have object files whose suffix is something other than ".o". This option allows you to specify another suffix, such as ".b" with *-o.b* or ":obj" with *-o:obj* and so forth.

–s *string* Starting string delimiter. This option permits you to specify a different string for **makedepend** to look for in the makefile.

–w *width* Line width. Normally, **makedepend** will ensure that every output line that it writes will be no wider than 78 characters for the sake of readability. This option enables you to change this width.

– – *options* – – If **makedepend** encounters a double hyphen (– –) in the argument list, then any unrecognized argument following it will be silently ignored; a second double hyphen terminates this special treatment. In this way, **makedepend** can be made to safely ignore esoteric compiler arguments that might normally be found in a CFLAGS **make** macro (see the **EXAMPLE** section above). All options that **makedepend** recognizes and appear between the pair of double hyphens are processed normally.

ALGORITHM

The approach used in this program enables it to run an order of magnitude faster than any other *dependency generator* that all files compiled by a single **makefile** will be compiled with roughly the same **-I** and **-D** options; and that most files in a single directory will include largely the same files.

Given these assumptions, **makedepend** expects to be called once for each **makefile**, with all source files that are maintained by the **makefile** appearing on the command line. It parses each source and include file exactly once, maintaining an internal symbol table for each. Thus, the first file on the command line will take an amount of time proportional to the amount of time that a normal C preprocessor takes. But on subsequent files, if it encounter's an include file that it has already parsed, it does not parse it again.

For example, imagine you are compiling two files, **file1.c** and **file2.c,** they each include the header file **header.h,** and the file **header.h** in turn includes the files **def1.h** and **def2.h.** When you run the command

```
makedepend file1.c file2.c
```

makedepend will parse **file1.c** and consequently, **header.h** and then **def1.h** and **def2.h.** It then decides that the dependencies for this file are

```
file1.o: header.h def1.h def2.h
```

But when the program parses **file2.c** and discovers that it, too, includes **header.h,** it does not parse the file, but simply adds **header.h, def1.h** and **def2.h** to the list of dependencies for **file2.o.**

SEE ALSO

cc(1B)**, make**(1S)

BUGS

If you do not have the source for **cpp**, the Berkeley C preprocessor, then **makedepend** will be compiled in such a way that all *#if* directives will evaluate to "true" regardless of their actual value. This may cause the wrong *#include* directives to be evaluated. **Makedepend** should simply have its own parser written for *#if* expressions.

Imagine you are parsing two files, **file1.c** and **file2.c,** each includes the file **def.h.** The list of files that **def.h** includes might truly be different when **def.h** is included by **file1.c** than when it is included by **file2.c.** But once **makedepend** arrives at a list of dependencies for a file, it is cast in concrete.

AUTHOR

Todd Brunhoff, Tektronix, Inc. and MIT Project Athena

NAME

makepsres – Build PostScript resource database file.

SYNOPSIS

makepsres [**–o** *filename*] [**–f** *filename*] [**–dir** *dirname*] [**–d**] [**–e**] [**–i**] [**–k**] [**–nb**] [**–p**] [**–q**] [**–s**] *directory ...*

DESCRIPTION

makepsres creates PostScript language resource database files. Resource database files can be used to specify the location of resources that are used by the font selection panel and other Adobe software. For a complete description of the resource location facilities in the Display PostScript system, see Appendix A and Appendix B of "Display PostScript Toolkit for X" in *Programming the Display PostScript System with X* .

makepsres creates a resource database file named **PSres.upr** that contains all the resources in all the *directory* path names specified on the command line.

If the list of directories contains – **,** **makepsres** reads from *stdin* and expects a list of directories separated by space, tab, or newline.

If the list of directories is empty, it is taken to be the current directory.

If all specified directories have a common initial prefix, **makepsres** extracts it as a directory prefix in the new resource database file.

makepsres normally acts recursively; it looks for resource files in subdirectories of any specified directory. This behavior can be overridden with the command line option –**nr.**

makepsres uses existing resource database files to assist in identifying files. By default, **makepsres** creates a new resource database file containing all of the following that apply:

Resource files found in the directories on the command line.

Resource files pointed to by the resource database files in the directories on the command line.

Resource entries found in the input resource database files. These entries are copied if the files they specify still exist and are located in directories not specified on the command line.

If you run **makepsres** in discard mode (with the –**d** option), it does not copy resource entries from the input resource database files. In that case, the output file consists only of entries from the directories on the command line. The input resource database files are only used to assist in identifying files.

If you run **makepsres** in keep mode (with the –**k** option), it includes in the output file all resource entries in the input resource database files, even entries for files that no longer exist or are located in directories specified on the command line.

makepsres uses various heuristics to identify files. A file that is of a private resource type or that does not conform to the standard format for a resource file must be specified in one of the following ways:

By running **makepsres** in interactive mode

By preloading the file into a resource database file used for input

By beginning the file with the following line:

%!PS-Adobe-3.0 Resource-<resource-type>

OPTIONS

–o *filename*
: Writes the output to the specified filename. The construction "**–o** –" writes to stdout. If the **–o** option is not specified, **makepsres** creates a **PSres.upr** file in the current directory and writes the output to that file.

–f *filename*
: Uses information from the specified file to assist in resource typing. The file must be in resource database file format. Multiple **–f** options may be specified. The construction "**–f** –" uses *stdin* as an input file and may not be used if "–" is specified as a directory on the command line.

–dir *dirname*
: Specifies that *dirname* is a directory. Needed only in rare cases when *dirname* is the same as a command-line option such as **–nb.**

–d Specifies discard mode. The resulting output file consists solely of entries from the directories on the command line.

–e Marks the resulting **PSres.upr** file as exclusive. This option makes the resource location library run more quickly since it does not have to look for other resource database files. It becomes necessary, however, to run **makepsres** whenever new resources are added to the directory, even if the resources come with their own resource database file.

–i Specifies interactive mode. In interactive mode, you will be queried for the resource type of any encountered file that **makepsres** cannot identify. If **–i** is not specified, **makepsres** assumes an unidentifiable file is not a resource file.

–k Specifies keep mode.

–nb If the output file already exists, do not back it up.

–nr Specifies nonrecursive mode. **makepsres** normally acts recursively: it looks for resource files in subdirectories of any specified directory. If **–nr** is used, **makepsres** does not look in subdirectories for resource files.

–p Specifies no directory prefix. If **–p** is used, **makepsres** does not try to find a common directory prefix among the specified directories.

–q Quiet mode: ignores unidentifiable files instead of warning about them.

–s Specifies strict mode. If **–s** is used, **makepsres** terminates with an error if it encounters a file it cannot identify.

EXAMPLES

example% makepsres .
: Creates a resource database file that contains all the resources in the current directory.

example% makepsres –i –o local.upr /usr/local/lib/ps/fonts
: Runs **makepsres** in interactive mode and creates a resource database file named **local.upr,** which contains all the resources in the directory **/usr/local/lib/ps/fonts.**

SEE ALSO

Programming the Display PostScript System with X (Addison-Wesley Publishing Company, Inc., 1993).

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

Copyright (c) 1989-1991 Adobe Systems Incorporated. All rights reserved.

NAME

maze – an automated maze program

SYNTAX

maze [**-S**] [**-r**] [**-g** *geometry*] [**-d** *display*]

DESCRIPTION

The **maze** program creates a "random" maze and then solves it with graphical feedback.

OPTIONS

-S	Full screen window option...
-r	Reverse video option...
-g *geometry*	Specifies the window geometry to be used...
-d *display*	Specifies the display to be used...

The following lists the current functionality of various mouse button clicks:

LeftButton	Clears the window and restarts maze...
MiddleButton	Toggles the maze program, first click -> *stop*, second click -> *continue*...
RightButton	Kills maze...

LIMITATIONS

No color support...
Expose events force a restart of maze...
Currently, mouse actions are based on "raw" values [Button1, Button2 and Button3] from the ButtonPress event...
[doesn't use pointer mapping]

COPYRIGHT

AUTHOR(s)

Richard Hess [X11 extensions] {...}!uunet!cimshop!rhess
 Consilium, Mountain View, CA
Dave Lemke [X11 version] lemke@sun.COM
 Sun MicroSystems, Mountain View, CA
Martin Weiss [SunView version]
 Sun MicroSystems, Mountain View, CA

NAME

mkdirhier – makes a directory hierarchy

SYNOPSIS

mkdirhier directory ...

DESCRIPTION

The **mkdirhier** command creates the specified directories. Unlike **mkdir** if any of the parent directories of the specified directory do not exist, it creates them as well.

SEE ALSO

mkdir(1)

NAME

mkfontdir – create fonts.dir file from directory of font files

SYNOPSIS

mkfontdir [*directory-names*]

DESCRIPTION

For each directory argument, **mkfontdir** reads all of the font files in the directory searching for properties named "FONT", or (failing that) the name of the file stripped of its suffix. These are used as font names, which are converted to lower case and written out to the file "fonts.dir" in the directory along with the name of the font file.

The kinds of font files read by mkfontdir depends on configuration parameters, but typically include PCF (suffix ".pcf"), SNF (suffix ".snf"), BDF (suffix ".bdf"), F3 (suffix ".f3b") and F3 bitmap (suffix ".fb"). If a font exists in multiple formats, **mkfontdir** will first choose fonts in fonts in the following order: F3, PCF, SNF, BDF and finally F3 bitmap (.fb).

SCALABLE FONTS

Because scalable font files do not usually include the X font name, the fonts.dir file in directories containing such fonts must be edited by hand to include the appropriate entries for those fonts. However, when **mkfontdir** is run, all of those additions will be lost, so be careful.

There is an alternative to editing "fonts.dir" file by hand. If there exists a "fonts.scale" file in any directory of the font-path, **mkfontdir** will copy the contents of this file to "fonts.dir" file. Any changes can therefore be safely made to "fonts.scale" and when **mkfontdir** is run in that directory, the changes will be reflected in the "fonts.dir" file also.

For example, the directory /usr/openwin/lib/X11/fonts/Speedo contains a "fonts.scale" file, which is hand created. The format of a "fonts.scale" file is as follows:

```
<num-of-entries>
filename        XLFD name
```

Note: F3 scalable fonts **do** contain enough information for **mkfontdir** to be able to construct the X font name (XLFD name). Therefore, the above does not apply to F3 format font files.

FONT NAME ALIASES

The file "fonts.alias" which can be put in any directory of the font-path is used to map new names to existing fonts, and should be edited by hand. The format is straight forward enough, two white-space separated columns, the first containing aliases and the second containing font-name patterns.

When a font alias is used, the name it references is search for in the normal manner, looking through each font directory in turn. This means that the aliases need not mention fonts in the same directory as the alias file.

To embed white-space in either name, simply enclose them in double-quote marks, to embed double-quote marks (or any other character), precede them with back-slash:

```
"magic-alias with spaces"      "\"font\name\" with quotes"
regular-alias                  fixed
```

If the string "FILE_NAMES_ALIASES" stands alone on a line, each file-name in the directory (stripped of it's suffix) will be used as an alias for that font.

USAGE

Both the **X server** and the **Font Server** look for "fonts.dir" and "fonts.alias" files in each directory in the font path each time it is set (see **xset**(1)).

SEE ALSO

X11(7), **Xserver**(1), **xset**(1)

NAME

muncher – draw interesting patterns in an X window

SYNOPSIS

muncher [**-r**] [**-s** *seed*] [**-v**] [**-q**] [**-geometry** *geometry*] [**-display** *display*]

OPTIONS

-r display in the root window

-s *seed* seed the random number seed

-v run in verbose mode

-q run in quite mode

-geometry *geometry*
define the initial window geometry; see **X11**(7)**.**

-display *display*
specify the display to use; see **X11**(7)**.**

DESCRIPTION

Muncher draws some interesting patterns in a window.

SEE ALSO

X11(7)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

NAME

oclock – display time of day

SYNOPSIS

oclock [-*options...*]

DESCRIPTION

Oclock simply displays the current time on an analog display

OPTIONS

-fg *foreground color*
choose a different color for the both hands and the jewel of the clock

-bg *background color*
choose a different color for the background.

-jewel *jewel color*
choose a different color for the jewel on the clock.

-minute *minute color*
choose a different color for the minute hand of the clock.

-hour *hour color*
choose a different color for the hour hand of the clock.

-backing WhenMapped | Always | NotUseful
selects an appropriate level of backing store.

-geometry *geometry*
define the initial window geometry; see **X11**(7)**.**

-display *display*
specify the display to use; see **X11**(7)**.**

-bd *border color*
choose a different color for the window border.

-bw *border width*
choose a different width for the window border. As the Clock widget changes its border around quite a bit, this is most usefully set to zero.

-noshape
causes the clock to not reshape itself and ancestors to exactly fit the outline of the clock.

-transparent
causes the clock to consist only of the jewel, the hands, and the border.

COLORS

If you would like your clock to be viewable in color, include the following in the #ifdef COLOR section you read with xrdb:

```
*customization:          -color
```

This will cause oclock to pick up the colors in the app-defaults color customization file: **/usr/openwin/lib/app-defaults/Clock-color**. Below are the default colors:

```
Clock*Background: grey
```

```
Clock∗BorderColor: light blue
Clock∗hour: yellow
Clock∗jewel: yellow
Clock∗minute: yellow
```

SEE ALSO

X11(7), X Toolkit documentation

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium

NAME

plaid – paint some plaid-like patterns in an X window

SYNOPSIS

plaid [**-b**] [**-fg** *color*] [**-bg** *color*] [**-bd** *color*] [**-bw** *number*] [**-geometry** *geometry*] [**-display** *display*]

OPTIONS

-b enable backing store for the window

-fg *color* This option specifies the color to use for the foreground of the window. The default is ‘‘white.’’

-bg *color*
This option specifies the color to use for the background of the window. The default is ‘‘black.’’

-bd *color*
This option specifies the color to use for the border of the window. The default is ‘‘white.’’

-bw *number*
This option specifies the width in pixels of the border surrounding the window.

-geometry *geometry*
define the initial window geometry; see **X11**(7)**.**

-display *display*
specify the display to use; see **X11**(7)**.**

DESCRIPTION

Plaid displays a continually changing plaid-like pattern in a window.

SEE ALSO

X11(7)

COPYRIGHT

NAME

pswrap – creates C procedures from segments of PostScript language code

SYNOPSIS

pswrap [–**apr**] [–**o** *outputCfile*] [–**h** *outputHfile*] [–**s** *maxstring*] *inputfile*

DESCRIPTION

pswrap reads input from *inputfile* and creates C-callable procedures, known as wraps, that send PostScript language code to the PostScript interpreter. *inputfile* contains segments of PostScript language code wrapped with a C-like procedure syntax.

Wraps are the most efficient way for an application to communicate with the PostScript interpreter. For complete documentation of **pswrap** and the language it accepts, see "pswrap Reference Manual" in *Programming the Display PostScript System with X*.

OPTIONS

inputfile
: A file that contains one or more wrap definitions. **pswrap** transforms the definitions in *inputfile* into C procedures. If no input file is specified, the standard input (which can be redirected from a file or pipe) is used. The input file can include text other than wrap definitions. **pswrap** converts wrap definitions to C procedures and passes the other text through unchanged. Therefore, it is possible to intersperse C-language source code with wrap definitions in the input file.

  Note: Although C code is allowed in a pswrap input file, it is not allowed within a wrap body. In particular, no CPP macros (for example, #define) are allowed inside a wrap.

–**a**
: Generates ANSI C procedure prototypes for procedure definitions in *outputCfile* and, optionally, *outputHfile.* The –**a** option allows compilers that recognize the ANSI C standard to do more complete type checking of parameters. The –**a** option also causes **pswrap** to generate const declarations.

  Note: ANSI C procedure prototype syntax is not recognized by most non-ANSI C compilers, including many compilers based on the Portable C Compiler. Use the –**a** option only in conjunction with a compiler that conforms to the ANSI C Standard.

–**h** *outputHFile*
: Generates a header file that contains extern declarations for non-static wraps. This file can be used in #include statements in modules that use wraps. If the –**a** option is specified, the declarations in the header file are ANSI C procedure prototypes. If the –**h** option is omitted, a header file is not produced.

–**o** *outputCFile*
: Specifies the file to which the generated wraps and passed-through text are written. If omitted, the standard output is used. If the –**a** option is also specified, the procedure definitions generated by **pswrap** are in ANSI C procedure prototype syntax.

–**p**
: Specifies that strings passed by wraps are padded so that each data object begins on a long-word (4-byte) boundary. This option allows wraps to run on architectures that restrict data alignment to 4-byte boundaries and improves

performance on some other architectures.

–r Generates reentrant code for wraps shared by more than one process (as in shared libraries). Reentrant code can be called recursively or by more than one thread. The **–r** option causes **pswrap** to generate extra code, so use it only when necessary.

–s *maxstring*
Sets the maximum allowable length of a PostScript string object or hexadecimal string object in the wrap body input. A syntax error is reported if a string is not terminated with) or > within *maxstring* characters. *maxstring* cannot be set lower than 80; the default is 200.

SEE ALSO

Programming the Display PostScript System with X (Addison-Wesley Publishing Company, Inc., 1993).

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

NAME

puzzle – 15-puzzle game for X

SYNOPSIS

puzzle [**-display** *display*] [**-geometry** *geometry*] [**-size** *WIDTH* x *HEIGHT*] [**-speed** *num*] [**-picture** *filename*] [**-colormap**]

OPTIONS

-display *display*
This option specifies the display to use; see **X11**(7).

-geometry *geometry*
This option specifies the size and position of the puzzle window; see **X11**(7).

-size *WIDTH* **x** *HEIGHT*
This option specifies the size of the puzzle in squares.

-speed *num*
This option specifies the speed in tiles per second for moving tiles around.

-picture *filename*
This option specifies an image file containing the picture to use on the tiles. Try ‘‘mandrill.cm.’’ This only works on 8-bit pseudo-color screens.

-colormap
This option indicates that the program should create its own colormap for the picture option.

DESCRIPTION

Puzzle with no arguments plays a 4x4 15-puzzle. The control bar has two boxes in it. Clicking in the left box scrambles the puzzle. Clicking in the right box solves the puzzle. Clicking the middle button anywhere else in the control bar causes puzzle to exit. Clicking in the tiled region moves the empty spot to that location if the region you click in is in the same row or column as the empty slot.

SEE ALSO

X11(7)

BUGS

The picture option should work on a wider variety of screens.

COPYRIGHT

Copyright 1988, Don Bennett.

AUTHOR

Don Bennett, HP Labs

NAME

ras2ps – converts a Sun RasterFile to a PostScript file

SYNOPSIS

ras2ps [–**x** *xoffset*] [–**y** *yoffset*] [–**X** *xscale*] [–**Y** *yscale*] [–**w** *width*] [–**h** *height*]
[–**r** *rotation*] [–**i**] [–**C**] [–**l**] [–**n**] [–**v**] [–**q**] [*rasterfile* | –] [*psfile*]

DESCRIPTION

ras2ps converts a Sun RasterFile to a PostScript file. If both filenames are missing, the rasterfile is read from **stdin** and the PostScript is written to **stdout**. If there is only one filename, then it is interpreted as the rasterfile and is opened for input. To have a named output PostScript file and still read the rasterfile from **stdin**, use a dash (-) in place of the input filename.

OPTIONS

–**x** *xoffset*
: Set the amount of space to the left of the image to be *xoffset* inches. The default is a quarter inch to keep the image in the imagable area of the printer.

–**y** *yoffset*
: Set the amount of space below the image to *yoffset* inches. The default is a quarter inch same as above.

–**X** *xscale*
: Multiply the width of the image by *xscale*. This is used to stretch or shrink an image along the X axis.

–**Y** *yscale*
: Multiply the height of the image by *yscale*. This is used to stretch or shrink an image along the Y axis.

–**w** *width*
: Set the max *width* in inches.

–**h** *height*
: Set the max *height* in inches. The -*w* and -*h* options set the desired width and height of the *output* image in inches. Default height and width are the source image dimensions at 300 dpi. **ras2ps** will expand or shrink the image to fit these dimensions, while still preserving the image scale values. The closest fit within the boundaries will be used. Note that width and height arguments do not stretch or shrink the image, but only set its limits. At least *one* of width or height is guaranteed to be satisfied.

–**r** *n* Rotate the image by *n* degrees, counterclockwise. The origin of rotation is the lower left corner of the image at the point specified by the -x and -y options.

–**i** Invert the image. This will reverse black and white on a monochrome image and is of limited usefulness on color images.

–**C** Output 8 and 24 bit images as color PostScript using the *colorimage* operator as supported by printers such as the *QMS ColorPS 800.*

–**l** Orient the image in landscape mode, which puts the origin at the lower right corner of the page and rotates the image 90 degrees. All arguments follow this new orientation. The default is Portrait mode.

−n Do not include the PostScript operator 'showpage' in the output. This is for backward compatibility with programs which do not override 'showpage' as the EPSF spec advises.

−v Verbose mode will print information as it processes the image. (The default is to be silent.)

−q Query (prints list of options)

SEE ALSO

lp(1), **24to8**(1)

NAME

rasterfile – Sun's file format for raster images

SYNOPSIS

#include <rasterfile.h>

DESCRIPTION

A rasterfile is composed of three parts: first, a header containing 8 integers; second, a (possibly empty) set of colormap values; and third, the pixel image, stored a line at a time, in increasing *y* order. The image is layed out in the file as in a memory pixrect. Each line of the image is rounded up to the nearest 16 bits.

The header is defined by the following structure:

```
struct rasterfile {
        int     ras_magic;
        int     ras_width;
        int     ras_height;
        int     ras_depth;
        int     ras_length;
        int     ras_type;
        int     ras_maptype;
        int     ras_maplength;
};
```

The *ras_magic* field always contains the following constant:

```
#define RAS_MAGIC    0x59a66a95
```

The *ras_width*, *ras_height*, and *ras_depth* fields contain the image's width and height in pixels, and its depth in bits per pixel, respectively. The depth is either 1 or 8, corresponding to standard frame buffer depths. The *ras_length* field contains the length in bytes of the image data. For an unencoded image, this number is computable from the *ras_width*, *ras_height*, and *ras_depth* fields, but for an encoded image it must be explicitly stored in order to be available without decoding the image itself. Note: the length of the header and of the (possibly empty) colormap values are not included in the value of the *ras_length* field; it is only the image data length. For historical reasons, files of type RT_OLD will usually have a 0 in the *ras_length* field, and software expecting to encounter such files should be prepared to compute the actual image data length if needed. The *ras_maptype* and *ras_maplength* fields contain the type and length in bytes of the colormap values, respectively. If *ras_maptype* is not RMT_NONE and the *ras_maplength* is not 0, then the colormap values are the *ras_maplength* bytes immediately after the header. These values are either uninterpreted bytes (usually with the *ras_maptype* set to RMT_RAW) or the equal length red, green and blue vectors, in that order (when the *ras_maptype* is RMT_EQUAL_RGB). In the latter case, the *ras_maplength* must be three times the size in bytes of any one of the vectors.

NAME

redxblue – swap red and blue for a 24 or 32 bit rasterfile

SYNOPSIS

redxblue [**-v**] [**-q**] [*inrasf*] [*outrasf*]

DESCRIPTION

redxblue converts an old-style 24 or 32 bit rasterfile into the newer, Sun-standard format. The old format had the byte ordering RGB for 24-bit rasterfiles and XRGB for 32-bit rasterfiles. The new format has the byte ordering XBGR for both 24-bit and 32-bit rasterfiles. X stands for undefined byte value.

The conversion is performed simply by swapping the red and blue bytes.

It is also possible to use this filter to swap 'red' and 'blue' bytes in any 32-bit rasterfile.

OPTIONS

–v Verbose mode will print information as it processes the image. (The default is to be silent.)

–q Query (prints list of options)

SEE ALSO

24to8(1)

NAME

resize – utility to set TERM and terminal settings to current window size

SYNOPSIS

resize [**-u** | **-c**] [**-s** [*row col*]]

DESCRIPTION

resize outputs a shell command for setting the **TERM** environment variable to indicate the current size of the **xterm**(1) window from which the command is run. For this output to take effect, **resize** must either be evaluated as part of the command line (usually done with a shell alias or function) or else redirected to a file which can then be read.

OPTIONS

The following options may be used with **resize**:

-u This option indicates that Bourne shell commands should be generated even if the user's current shell isn't **/bin/sh**.

-c This option indicates that C shell commands should be generated even if the user's current shell isn't **/bin/csh**.

-s [*row col*]
This option indicates that that Sun console escape sequences will be used instead of the special **xterm** escape code. If *row* and *col* are given, **resize** will ask the **xterm** to resize itself. However, the window manager may choose to disallow the change.

EXAMPLES

From the C shell (usually known as **/bin/csh**), the following alias could be defined in the user's **.cshrc**:

```
example% alias rs 'set noglob; ‘eval resize‘'
```

After resizing the window, the user would type:

```
example% rs
```

Users of versions of the Bourne shell (usually known as **/bin/sh**) that don't have command functions will need to send the output to a temporary file and the read it back in with the ‘‘.’’ command:

```
$ resize >/tmp/out
$ . /tmp/out
```

FILES

/usr/share/lib/termcap
/usr/share/lib/terminfo/?/*
~/.cshrc

SEE ALSO

csh(1), **tset**(1B), **xterm**(1)

AUTHORS

Mark Vandevoorde (MIT-Athena), Edward Moy (Berkeley)
Copyright (c) 1984, 1985 by Massachusetts Institute of Technology.
See **X11**(7) for a complete copyright notice.

BUGS

The **-u** or **-c** must appear to the left of **-s** if both are specified.

There should be some global notion of display size; **terminfo** needs to be rethought in the context of window systems. (Fixed in 4.3BSD, and Ultrix-32 1.2)

NAME

rgb – build the color name database

SYNOPSIS

rgb [*dbname*]

DESCRIPTION

rgb reads from standard input lines of the form:

red green blue name

where *red* / *green* / *blue* are decimal values between the range 0 to 255, and *name* is a description of the color. **rgb** then builds a color name database in **dbm** format. The color name database provides a mapping between ASCII color names and RGB color values. It is useful for increasing the portability of color programs. The input source for the database is in **/usr/openwin/lib/rgb.txt**. **rgb.txt** is compiled into the dbm files **rgb.dir** and **rgb.pag**. When the server first starts up, it consults the contents of these files to build an internal representation of their contents. This internal representation is consulted to map color names to color values.

rgb uses the default color name database of **/usr/openwin/lib/X11/rgb.txt**.

OPTIONS

dbname	Color name database.

EXAMPLES

```
example% rgb rgb < rgb.txt
example% cat /usr/openwin/lib/rgb.txt
```

FILES

/usr/openwin/lib/rgb.txt	color name database source. Maps color names to RBG color values.
/usr/openwin/lib/rgb.dir	dbm file containing color name to RGB mapping.
/usr/openwin/lib/rgb.pag	dbm file containing the color name to RGB mapping.

SEE ALSO

bldrgb(1), **cat**(1), **dbm**(3)

NAME

sessreg – manage utmp / wtmp entries for non-init clients

SYNOPSIS

sessreg [-w *wtmp-file*] [-u *utmp-file*] [-l *line-name*] [-h *host-name*] [-s *slot-number*] [-x *Xservers-file*] [-t *ttys-file*] [-a] [-d] *user-name*

DESCRIPTION

Sessreg is a simple program for managing utmp / wtmp entries for **xdm**(1) sessions.

System V has a better interface to /etc/utmp than BSD; it dynamically allocates entries in the file, instead of writing them at fixed positions indexed by position in /etc/ttys.

To manage BSD-style utmp files, **sessreg** has two strategies. In conjunction with **xdm**(1), the -x option counts the number of lines in /etc/ttys and then adds to that the number of the line in the Xservers file which specifies the display. The display name must be specified as the "line-name" using the -l option. This sum is used as the "slot-number" in /etc/utmp that this entry will be written at. In the more general case, the -s option specifies the slot-number directly. If for some strange reason your system uses a file other that /etc/ttys to manage init, the -t option can direct **sessreg** to look elsewhere for a count of terminal sessions.

Conversely, System V managers will not ever need to use these options (-x, -s and -t). To make the program easier to document and explain, **sessreg** accepts the BSD-specific flags in the System V environment and ignores them.

BSD also has a host-name field in the utmp file which doesn't exist in System V. This option is also ignored by the System V version of **sessreg.**

USAGE

In Xstartup, place a call like:

```
sessreg -a -l $DISPLAY -x /usr/openwin/lib/X11/xdm/Xservers $USER
```

and in Xreset:

```
sessreg -d -l $DISPLAY -x /usr/openwin/lib/X11/xdm/Xservers $USER
```

OPTIONS

-w *wtmp-file*
: This specifies an alternate wtmp file, instead of /usr/adm/wtmp for BSD or /etc/wtmp for sysV. The special name "none" disables writing records to /usr/adm/wtmp.

-u *utmp-file*
: This specifies an alternate utmp file, instead of "/etc/utmp". The special name "none" disables writing records to /etc/utmp.

-l *line-name*
: This describes the "line" name of the entry. For terminal sessions, this is the final pathname segment of the terminal device filename (e.g. ttyd0). For X sessions, it should probably be the local display name given to the users session (e.g. :0). If none is specified, the terminal name will be determined with ttyname(3) and stripped of leading components.

-h *host-name*
: This is set for BSD hosts to indicate that the session was initiated from a remote

host. In typical xdm usage, this options is not used.

-s *slot-number*
: Each potential session has a unique slot number in BSD systems, most are identified by the position of the *line-name* in the /etc/ttys file. This option overrides the default position determined with **ttyslot**(3C). This option is inappropriate for use with xdm, the -x option is more useful.

-x *Xservers-file*
: As X sessions are one-per-display, and each display is entered in this file, this options sets the *slot-number* to be the number of lines in the *ttys-file* plus the index into this file that the *line-name* is found.

-t *ttys-file*
: This specifies an alternate file which the *-x* option will use to count the number of terminal sessions on a host.

-a This session should be added to utmp / wtmp.

-d This session should be deleted from utmp / wtmp. One of -a / -d must be specified.

SEE ALSO

xdm(1)

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium

NAME

showfont – font dumper for X font server

SYNOPSIS

showfont [–**bitmap_pad#**] [–**end***#*] [–**extents_only**] [–**fn** *pattern*] [–**l**] [–**L**] [–**m**] [–**M**] [–**server** *host:port*] [–**start***#*] [–**unit***#*]

DESCRIPTION

showfont displays data about the font that matches the given *pattern*. The wildcard character "∗" may be used to match any sequence of characters (including none), and "?" to match any single character. If no *pattern* is given, "∗" is assumed. The "∗" and "?" characters must be quoted to prevent them from being expanded by the shell.

OPTIONS

–**server** *host:port*
: This option specifies the X font server to contact.

–**l** This option indicates that the bit order of the font is LSBFirst (least significant bit first).

–**m** This option indicates that the bit order of the font is MSBFirst (most significant bit first).

–**L** This option indicates that the byte order of the font is LSBFirst (least significant byte first).

–**M** This option indicates that the byte order of the font is MSBFirst (most significant byte first).

–**extents_only**
: This option indicates that only the font's extents should be displayed.

–**start***#* This option indicates the start of the range of the characters to display (# is a number).

–**end***#* This option indicates the end of the range of the characters to display (# is a number).

–**pad***#* This option specifies the scanpad unit of the font (1, 2, 4 or 8). (# is a number).

–**bitmap_pad#**
: This option specifies the bitmap padding unit of the font (0, 1, or 2, where 0 is *ImageRectMin*, 1 is *ImageRectMaxWidth* and 2 is *ImageRectMaxWidth*).

SEE ALSO

fs(1), **showfont**(1), **xlsfonts**(1)

ENVIRONMENT

FONTSERVER
: to get the default host and port to use.

COPYRIGHT

AUTHOR

Dave Lemke, Network Computing Devices, Inc

NAME

showrgb – display the color name database

SYNOPSIS

showrgb [**dbname**]

DESCRIPTION

showrgb displays the contents of the **dbm** format color name database. The color name database provides a mapping between ASCII color names and RGB color values. It is useful for increasing the portability of color programs. The source of the database is in **/usr/openwin/lib/X11/rgb.txt**. **rgb.txt** is compiled into the dbm database files **rgb.dir** and **rgb.pag.** When the server first starts up, it consults the contents of these files to build an internal representation of their contents. This internal representation is consulted to map color names to color values. **showrgb** is useful in debugging color name databases to guarantee that the binary file is representative of the ASCII source file.

showrgb uses the default color name database of **/usr/openwin/lib/X11/rgb.txt**.

OPTIONS

dbname	Color name database.

EXAMPLES

```
example% showrgb /usr/openwin/lib/rgb
example% cat /usr/openwin/lib/rgb.txt
```

FILES

/usr/openwin/lib/rgb.txt	color name database source. Maps color names to RGB color values.
/usr/openwin/lib/rgb.dir	dbm file containing color name to RGB mapping.
/usr/openwin/lib/rgb.pag	dbm file containing color name to RGB mapping.

SEE ALSO

bldrgb(1), **cat**(1), **rgb**(1), **dbm**(3)

NAME

showsnf – display contents of an SNF file

SYNOPSIS

showsnf [**-v**] [**-g**] [**-m**] [**-M**] [**-l**] [**-L**] [**-p**#] [**-u**#]

DESCRIPTION

The **showsnf** utility displays the contents of font files in the Server Natural Format produced by **bsdtosnf**. It is usually only to verify that a font file hasn't been corrupted or to convert the individual glyphs into arrays of characters for proofreading or for conversion to some other format.

OPTIONS

–v This option indicates that character bearings and sizes should be printed.

–g This option indicates that character glyph bitmaps should be printed.

–m This option indicates that the bit order of the font is MSBFirst (most significant bit first).

–l This option indicates that the bit order of the font is LSBFirst (least significant bit first).

–M This option indicates that the byte order of the font is MSBFirst (most significant byte first).

–L This option indicates that the byte order of the font is LSBFirst (least significant byte first).

–p# This option specifies the glyph padding of the font (# is a number).

–u# This option specifies the scanline unit of the font (# is a number).

SEE ALSO

bdftosnf(1)

BUGS

There is no way to just print out a single glyph.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

NAME

texteroids – test your mousing skills on spinning text

SYNOPSIS

texteroids [**–display** *name*][**–fn** *font*][**–size** *size*][*text_string*]

DESCRIPTION

texteroids spins the specified text string in a window. If you click on the text with the mouse, the string splits up into individual letters, each of which you may then click on.

OPTIONS

–display *name*
specifies the display on which to open a connection to the Display PostScript System. If no display is specified, the DISPLAY environment variable is used.

–fn *font*
specifies the name of the PostScript language font software to use. The default is Times-Italic.

–size *size*
specifies the size, in points, of the text. The default is 36.

text_string
specifies the text to display. If the text has spaces it must be enclosed in quotation marks. The default text string is "Adobe".

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

NAME

twm – Tab Window Manager for the X Window System

SYNOPSIS

twm [**-display** *dpy*] [**-s**] [**-f** *initfile*] [**-v**]

DESCRIPTION

The **twm** program is a window manager for the X Window System. It provides titlebars, shaped windows, several forms of icon management, user-defined macro functions, click-to-type and pointer-driven keyboard focus, and user-specified key and pointer button bindings.

This program is usually started by the user's session manager or startup script. When used from **xdm**(1) or **xinit**(1) without a session manager, **twm** is frequently executed in the foreground as the last client. When run this way, exiting **twm** causes the session to be terminated (i.e. logged out).

By default, application windows are surrounded by a ''frame'' with a titlebar at the top and a special border around the window. The titlebar contains the window's name, a rectangle that is lit when the window is receiving keyboard input, and function boxes known as ''titlebuttons'' at the left and right edges of the titlebar.

Pressing pointer Button1 (usually the left-most button unless it has been changed with **xmodmap**(1)) on a titlebutton will invoke the function associated with the button. In the default interface, windows are iconified by clicking (pressing and then immediately releasing) the left titlebutton (which looks like a Dot). Conversely, windows are deiconified by clicking in the associated icon or entry in the icon manager (see description of the variable **ShowIconManager** and of the function **f.showiconmgr**).

Windows are resized by pressing the right titlebutton (which resembles a group of nested squares), dragging the pointer over edge that is to be moved, and releasing the pointer when the outline of the window is the desired size. Similarly, windows are moved by pressing in the title or highlight region, dragging a window outline to the new location, and then releasing when the outline is in the desired position. Just clicking in the title or highlight region raises the window without moving it.

When new windows are created, **twm** will honor any size and location information requested by the user (usually through **-geometry** command line argument or resources for the individual applications). Otherwise, an outline of the window's default size, its titlebar, and lines dividing the window into a 3x3 grid that track the pointer are displayed. Clicking pointer Button1 will position the window at the current position and give it the default size. Pressing pointer Button2 (usually the middle pointer button) and dragging the outline will give the window its current position but allow the sides to be resized as described above. Clicking pointer Button3 (usually the right pointer button) will give the window its current position but attempt to make it long enough to touch the bottom the screen.

OPTIONS

twm accepts the following command line options:

–display *dpy*
    This option specifies the X server to use.

–s This option indicates that only the default screen (as specified by **–display** or by the **DISPLAY** environment variable) should be managed. By default, **twm** will attempt to manage all screens on the display.

–f *filename*
This option specifies the name of the startup file to use. By default, **twm** will look in the user's home directory for files named **.twmrc.***num* **(where** *num* **is a screen**

–v This option indicates that **twm** should print error messages whenever an unexpected X Error event is received. This can be useful when debugging applications but can be distracting in regular use.

CUSTOMIZATION

Much of **twm**'s appearance and behavior can be controlled by providing a startup file in one of the following locations (searched in order for each screen being managed when **twm** begins):

$HOME/.twmrc.*screennumber*
The *screennumber* is a small positive number (e.g. 0, 1, etc.) representing the screen number (e.g. the last number in the **DISPLAY** environment variable *host:displaynum.screennum*) that would be used to contact that screen of the display. This is intended for displays with multiple screens of differing visual types.

$HOME/.twmrc
This is the usual name for an individual user's startup file.

/usr/openwin/lib/X11/system.twmrc
If neither of the preceding files are found, **twm** will look in this file for a default configuration. This is often tailored by the site administrator to provide convenient menus or familiar bindings for novice users.

If no startup files are found, **twm** will use the built-in defaults described above. The only resource used by **twm** is **bitmapFilePath** for a colon-separated list of directories to search when looking for bitmap files (for more information, see the *Athena Widgets* manual and **xrdb**(1)).

twm startup files are logically broken up into three types of specifications: **Variables**, **Bindings**, **Menus**. The **Variables** section must come first and is used to describe the fonts, colors, cursors, border widths, icon and window placement, highlighting, autoraising, layout of titles, warping, use of the icon manager. The **Bindings** section usually comes second and is used to specify the functions that should be to be invoked when keyboard and pointer buttons are pressed in windows, icons, titles, and frames. The **Menus** section gives any user-defined menus (containing functions to be invoked or commands to be executed).

Variable names and keywords are case-insensitive. Strings must be surrounded by double quote characters (e.g. "blue") and are case-sensitive. A pound sign (#) outside of a string causes the remainder of the line in which the character appears to be treated as a comment.

VARIABLES

Many of the aspects of **twm**'s user interface are controlled by variables that may be set in the user's startup file. Some of the options are enabled or disabled simply by the presence of a particular keyword. Other options require keywords, numbers, strings, or lists of all of these.

Lists are surrounded by braces and are usually separated by whitespace or a newline. For example:

```
AutoRaise { "emacs" "XTerm" "Xmh" }
```

or

```
AutoRaise
{
        "emacs"
        "XTerm"
        "Xmh"
}
```

When a variable containing a list of strings representing windows is searched (e.g. to determine whether or not to enable autoraise as shown above), a string must be an exact, case-sensitive match to the window's name name (given by the WM_NAME window property), resource name or class name (both given by the WM_CLASS window property). The preceding example would enable autoraise on windows named ‘‘emacs’’ as well as any **xterm**(1) (since they are of class ‘‘XTerm’’) or xmh windows (which are of class ‘‘Xmh’’).

String arguments that are interpreted as filenames (see the **Pixmaps**, **Cursors**, and **IconDirectory** below) will prepend the user's directory (specified by the **HOME** environment variable) if the first character is a tilde (˜). If, instead, the first character is a colon (:), the name is assumed to refer to one of the internal bitmaps that are used to create the default titlebars symbols: **:xlogo** or **:iconify** (both refer to the X used for the iconify button), **:resize** (the nested squares used by the resize button), and **:question** (the question mark used for non-existent bitmap files).

The following variables may be specified at the top of a **twm** startup file. Lists of Window name prefix strings are indicated by *win-list*. Optional arguments are shown in square brackets:

AutoRaise { *win-list* }
: This variable specifies a list of windows that should automatically be raised whenever the pointer enters the window. This action can be interactively enabled or disabled on individual windows using the function **f.autoraise**.

AutoRelativeResize
: This variable indicates that dragging out a window size (either when initially sizing the window with pointer Button2 or when resizing it) should not wait until the pointer has crossed the window edges. Instead, moving the pointer automatically causes the nearest edge or edges to move by the same amount. This allows the resizing of windows that extend off the edge of the screen. If the pointer is in the center of the window, or if the resize is begun by pressing a

titlebutton, **twm** will still wait for the pointer to cross a window edge (to prevent accidents). This option is particularly useful for people who like the press-drag-release method of sweeping out window sizes.

BorderColor *string* [{ *wincolorlist* }]
This variable specifies the default color of the border to be placed around all non-iconified windows, and may only be given within a **Color** or **Monochrome** list. The optional *wincolorlist* specifies a list of window and color name pairs for specifying particular border colors for different types of windows. For example:

```
BorderColor "gray50"
{
        "XTerm"         "red"
        "xmh"   "green"
}
```

The default is "black".

BorderTileBackground *string* [{ *wincolorlist* }]
This variable specifies the default background color in the gray pattern used in unhighlighted borders (only if **NoHighlight** hasn't been set), and may only be given within a **Color** or **Monochrome** list. The optional *wincolorlist* allows per-window colors to be specified. The default is "white".

BorderTileForeground *string* [{ *wincolorlist* }]
This variable specifies the default foreground color in the gray pattern used in unhighlighted borders (only if **NoHighlight** hasn't been set), and may only be given within a **Color** or **Monochrome** list. The optional *wincolorlist* allows per-window colors to be specified. The default is "black".

BorderWidth *pixels*
This variable specifies the width in pixels of the border surrounding all client window frames if **ClientBorderWidth** has not been specified. This value is also used to set the border size of windows created by **twm** (such as the icon manager). The default is 2.

ButtonIndent *pixels*
This variable specifies the amount by which titlebuttons should be indented on all sides. Positive values cause the buttons to be smaller than the window text and highlight area so that they stand out. Setting this and the **TitleButtonBorderWidth** variables to 0 makes titlebuttons be as tall and wide as possible. The default is 1.

ClientBorderWidth
This variable indicates that border width of a window's frame should be set to the initial border width of the window, rather than to the value of **BorderWidth**.

Color { *colors-list* }
This variable specifies a list of color assignments to be made if the default display is capable of displaying more than simple black and white. The *colors-list* is made up of the following color variables and their values:

DefaultBackground, **DefaultForeground**, **MenuBackground**, **MenuForeground**, **MenuTitleBackground**, **MenuTitleForeground**, and **MenuShadowColor**. The following color variables may also be given a list of window and color name pairs to allow per-window colors to be specified (see **BorderColor** for details): **BorderColor**, **IconManagerHighlight**, **BorderTitleBackground**, **BorderTitleForeground**, **TitleBackground**, **TitleForeground**, **IconBackground**, **IconForeground**, **IconBorderColor**, **IconManagerBackground**, and **IconManagerForeground**. For example:

```
Color
{
        MenuBackground          "gray50"
        MenuForeground          "blue"
        BorderColor             "red" { "XTerm" "yellow" }
        TitleForeground         "yellow"
        TitleBackground         "blue"
}
```

All of these color variables may also be specified for the **Monochrome** variable, allowing the same initialization file to be used on both color and monochrome displays.

ConstrainedMoveTime *milliseconds*
This variable specifies the length of time between button clicks needed to begin a constrained move operation. Double clicking within this amount of time when invoking **f.move** will cause the window only be moved in a horizontal or vertical direction. Setting this value to 0 will disable constrained moves. The default is 400 milliseconds.

Cursors { *cursor-list* }
This variable specifies the glyphs that **twm** should use for various pointer cursors. Each cursor may be defined either from the **cursor** font or from two bitmap files. Shapes from the **cursor** font may be specified directly as:

cursorname "*string*"

where *cursorname* is one of the cursor names listed below, and *string* is the name of a glyph as found in the file /usr/include/X11/cursorfont.h (without the ‘‘XC_’’ prefix). If the cursor is to be defined from bitmap files, the following syntax is used instead:

cursorname "*image*" "*mask*"

The *image* and *mask* strings specify the names of files containing the glyph image and mask in **bitmap**(1) form. The bitmap files are located in the same manner as icon bitmap files. The following example shows the default cursor definitions:

```
Cursors
{
        Frame           "top_left_arrow"
        Title           "top_left_arrow"
        Icon            "top_left_arrow"
        IconMgr         "top_left_arrow"
        Move            "fleur"
        Resize          "fleur"
        Menu            "sb_left_arrow"
        Button          "hand2"
        Wait            "watch"
        Select          "dot"
        Destroy"pirate"
}
```

DecorateTransients
This variable indicates that transient windows (those containing a WM_TRANSIENT_FOR property) should have titlebars. By default, transients are not reparented.

DefaultBackground *string*
This variable specifies the background color to be used for sizing and information windows. The default is "white".

DefaultForeground *string*
This variable specifies the foreground color to be used for sizing and information windows. The default is "black".

DontIconifyByUnmapping { *win-list* }
This variable specifies a list of windows that should not be iconified by simply unmapping the window (as would be the case if **IconifyByUnmapping** had been set). This is frequently used to force some windows to be treated as icons while other windows are handled by the icon manager.

DontMoveOff
This variable indicates that windows should not be allowed to be moved off the screen. It can be overridden by the **f.forcemove** function.

DontSqueezeTitle [{ *win-list* }]
This variable indicates that titlebars should not be squeezed to their minimum size as described under **SqueezeTitle** below. If the optional window list is supplied, only those windows will be prevented from being squeezed.

ForceIcons
This variable indicates that icon pixmaps specified in the **Icons** variable should override any client-supplied pixmaps.

FramePadding *pixels*
This variable specifies the distance between the titlebar decorations (the button

and text) and the window frame. The default is 2 pixels.

IconBackground *string* [{ *win-list* }]
This variable specifies the background color of icons, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **Border-Color** variable for a complete description of the *win-list*. The default is "white".

IconBorderColor *string* [{ *win-list* }]
This variable specifies the color of the border used for icon windows, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "black".

IconBorderWidth *pixels*
This variable specifies the width in pixels of the border surrounding icon windows. The default is 2.

IconDirectory *string*
This variable specifies the directory that should be searched if if a bitmap file cannot be found in any of the directories in the **bitmapFilePath** resource.

IconFont *string*
This variable specifies the font to be used to display icon names within icons. The default is "variable".

IconForeground *string* [{ *win-list* }]
This variable specifies the foreground color to be used when displaying icons, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "black".

IconifyByUnmapping [{ *win-list* **}]**
This variable indicates that windows should be iconified by being unmapped without trying to map any icons. This assumes that the user will remap the window through the icon manager, the **f.warpto** function, or the *TwmWindows* menu. If the optional *win-list* is provided, only those windows will be iconified by simply unmapping. Windows that have both this and the **IconManagerDontShow** options set may not be accessible if no binding to the *TwmWindows* menu is set in the user's startup file.

IconManagerBackground *string* [{ *win-list* }]
This variable specifies the background color to use for icon manager entries, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "white".

IconManagerDontShow [{ *win-list* }]
This variable indicates that the icon manager should not display any windows.

If the optional *win-list* is given, only those windows will not be displayed. This variable is used to prevent windows that are rarely iconified (such as **xclock**(1) or **xload**(1)) from taking up space in the icon manager.

IconManagerFont *string*
This variable specifies the font to be used when displaying icon manager entries. The default is "variable".

IconManagerForeground *string* [{ *win-list* }]
This variable specifies the foreground color to be used when displaying icon manager entries, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list.* The default is "black".

IconManagerGeometry *string* [*columns*]
This variable specifies the geometry of the icon manager window. The *string* argument is standard geometry specification that indicates the initial full size of the icon manager. The icon manager window is then broken into *columns* pieces and scaled according to the number of entries in the icon manager. Extra entries are wrapped to form additional rows. The default number of columns is 1.

IconManagerHighlight *string* [{ *win-list* }]
This variable specifies the border color to be used when highlighting the icon manager entry that currently has the focus, and can only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list.* The default is "black".

IconManagers { *iconmgr-list* }
This variable specifies a list of icon managers to create. Each item in the *iconmgr-list* has the following format:

"*winname*" ["*iconname*"] "*geometry*" *columns*

where *winname* is the name of the windows that should be put into this icon manager, *iconname* is the name of that icon manager window's icon, *geometry* is a standard geometry specification, and *columns* is the number of columns in this icon manager as described in **IconManagerGeometry**. For example:

```
IconManagers
{
        "XTerm"         "=300x5+800+5" 5
        "myhost"        "=400x5+100+5" 2
}
```

Clients whose name or class is ‘‘XTerm’’ will have an entry created in the ‘‘XTerm’’ icon manager. Clients whose name was ‘‘myhost’’ would be put into the ‘‘myhost’’ icon manager.

IconManagerShow { *win-list* }

This variable specifies a list of windows that should appear in the icon manager. When used in conjunction with the **IconManagerDontShow** variable, only the windows in this list will be shown in the icon manager.

IconRegion *geomstring vgrav hgrav gridwidth gridheight*

This variable specifies an area on the root window in which icons are placed if no specific icon location is provided by the client. The *geomstring* is a quoted string containing a standard geometry specification. If more than one **IconRegion** lines are given, icons will be put into the succeeding icon regions when the first is full. The **vgrav** argument should be either **North** or **South** and control and is used to control whether icons are first filled in from the top or bottom of the icon region. Similarly, the *hgrav* argument should be either **East** or **West** and is used to control whether icons should be filled in from left from the right. Icons are laid out within the region in a grid with cells *gridwidth* pixels wide and *gridheight* pixels high.

Icons { *win-list* }

This variable specifies a list of window names and the bitmap filenames that should be used as their icons. For example:

```
Icons
{
        "XTerm"         "xterm.icon"
        "xfd"           "xfd_icon"
}
```

Windows that match ‘‘XTerm’’ and would not be iconified by unmapping, and would try to use the icon bitmap in the file ‘‘xterm.icon’’. If **ForceIcons** is specified, this bitmap will be used even if the client has requested its own icon pixmap.

InterpolateMenuColors

This variable indicates that menu entry colors should be interpolated between entry specified colors. In the example below:

```
Menu "mymenu"
{
        "Title"         ("black":"red")         f.title
        "entry1"                        f.nop
        "entry2"                        f.nop
        "entry3"("white":"green")       f.nop
        "entry4"                        f.nop
        "entry5"("red":"white")         f.nop
}
```

the foreground colors for ‘‘entry1’’ and ‘‘entry2’’ will be interpolated between black and white, and the background colors between red and green. Similarly, the foreground for ‘‘entry4’’ will be half-way between white and red, and the background will be half-way between green and white.

MakeTitle { *win-list* }
This variable specifies a list of windows on which a titlebar should be placed and is used to request titles on specific windows when **NoTitle** has been set.

MaxWindowSize *string*
This variable specifies a geometry in which the width and height give the maximum size for a given window. This is typically used to restrict windows to the size of the screen. The default is "30000x30000".

MenuBackground *string*
This variable specifies the background color used for menus, and can only be specified inside of a **Color** or **Monochrome** list. The default is "white".

MenuFont *string*
This variable specifies the font to use when displaying menus. The default is "variable".

MenuForeground *string*
This variable specifies the foreground color used for menus, and can only be specified inside of a **Color** or **Monochrome** list. The default is "black".

MenuShadowColor *string*
This variable specifies the color of the shadow behind pull-down menus and can only be specified inside of a **Color** or **Monochrome** list. The default is "black".

MenuTitleBackground *string*
This variable specifies the background color for **f.title** entries in menus, and can only be specified inside of a **Color** or **Monochrome** list. The default is "white".

MenuTitleForeground *string*
This variable specifies the foreground color for **f.title** entries in menus and can only be specified inside of a **Color** or **Monochrome** list. The default is "black".

Monochrome { *colors* }
This variable specifies a list of color assignments that should be made if the screen has a depth of 1. See the description of **Colors**.

MoveDelta *pixels*
This variable specifies the number of pixels the pointer must move before the **f.move** function starts working. Also see the **f.deltastop** function. The default is zero pixels.

NoBackingStore
This variable indicates that **twm**'s menus should not request backing store to minimize repainting of menus. This is typically used with servers that can repaint faster than they can handle backing store.

NoCaseSensitive
This variable indicates that case should be ignored when sorting icon names in an icon manager. This option is typically used with applications that capitalize the first letter of their icon name.

NoDefaults
This variable indicates that **twm** should not supply the default titlebuttons and

bindings. This option should only be used if the startup file contains a completely new set of bindings and definitions.

NoGrabServer
: This variable indicates that **twm** should not grab the server when popping up menus and moving opaque windows.

NoHighlight [{ *win-list* }]
: This variable indicates that borders should not be highlighted to track the location of the pointer. If the optional *win-list* is given, highlighting will only be disabled for those windows. When the border is highlighted, it will be drawn in the current **BorderColor**. When the border is not highlighted, it will be stippled with an gray pattern using the current **BorderTileForeground** and **BorderTileBackground** colors.

NoIconManagers
: This variable indicates that no icon manager should be created.

NoMenuShadows
: This variable indicates that menus should not have drop shadows drawn behind them. This is typically used with slower servers since it speeds up menu drawing at the expense of making the menu slightly harder to read.

NoRaiseOnDeiconify
: This variable indicates that windows that are deiconified should not be raised.

NoRaiseOnMove
: This variable indicates that windows should not be raised when moved. This is typically used to allow windows to slide underneath each other.

NoRaiseOnResize
: This variable indicates that windows should not be raised when resized. This is typically used to allow windows to be resized underneath each other.

NoRaiseOnWarp
: This variable indicates that windows should not be raised when the pointer is warped into them with the **f.warpto** function. If this option is set, warping to an occluded window may result in the pointer ending up in the occluding window instead the desired window (which causes unexpected behavior with **f.warpring**).

NoSaveUnders
: This variable indicates that menus should not request save-unders to minimize window repainting following menu selection. It is typically used with displays that can repaint faster than they can handle save-unders.

NoStackMode [{ *win-list* }]
: This variable indicates that client window requests to change stacking order should be ignored. If the optional *win-list* is given, only requests on those windows will be ignored. This is typically used to prevent applications from relentlessly popping themselves to the front of the window stack.

NoTitle [{ *win-list* }]

This variable indicates that windows should not have titlebars. If the optional *win-list* is given, only those windows will not have titlebars. **MakeTitle** may be used with this option to force titlebars to be put on specific windows.

NoTitleFocus

This variable indicates that **twm** should not set keyboard input focus to each window as it is entered. Normally, **twm** sets the focus so that focus and key events from the titlebar and icon managers are delivered to the application. If the pointer is moved quickly and **twm** is slow to respond, input can be directed to the old window instead of the new. This option is typically used to prevent this ‘‘input lag’’ and to work around bugs in older applications that have problems with focus events.

NoTitleHighlight [{ *win-list* }]

This variable indicates that the highlight area of the titlebar, which is used to indicate the window that currently has the input focus, should not be displayed. If the optional *win-list* is given, only those windows will not have highlight areas. This and the **SqueezeTitle** options can be set to substantially reduce the amount of screen space required by titlebars.

OpaqueMove

This variable indicates that the **f.move** function should actually move the window instead of just an outline so that the user can immediately see what the window will look like in the new position. This option is typically used on fast displays (particularly if **NoGrabServer** is set).

Pixmaps { *pixmaps* }

This variable specifies a list of pixmaps that define the appearance of various images. Each entry is a keyword indicating the pixmap to set, followed by a string giving the name of the bitmap file. The following pixmaps may be specified:

```
Pixmaps
{
        TitleHighlight    "gray1"
}
```

The default for *TitleHighlight* is to use an even stipple pattern.

RandomPlacement

This variable indicates that windows with no specified geometry should should be placed in a pseudo-random location instead of having the user drag out an outline.

ResizeFont *string*

This variable specifies the font to be used for in the dimensions window when resizing windows. The default is "fixed".

RestartPreviousState

This variable indicates that **twm** should attempt to use the WM_STATE property on client windows to tell which windows should be iconified and which should be left visible. This is typically used to try to regenerate the state that the screen was in before the previous window manager was shutdown.

SaveColor { *colors-list* }

This variable indicates a list of color assignments to be stored as pixel values in the root window property _MIT_PRIORITY_COLORS. Clients may elect to preserve these values when installing their own colormap. Note that use of this mechanism is a way an for application to avoid the "technicolor" problem, whereby useful screen objects such as window borders and titlebars disappear when a programs custom colors are installed by the window manager. For example:

```
SaveColor
{
    BorderColor
    TitleBackground
    TitleForeground
    "red"
    "green"
    "blue"
}
```

This would place on the root window 3 pixel values for borders and titlebars, as well as the three color strings, all taken from the default colormap.

ShowIconManager

This variable indicates that the icon manager window should be displayed when **twm** is started. It can always be brought up using the **f.showiconmgr** function.

SortIconManager

This variable indicates that entries in the icon manager should be sorted alphabetically rather than by simply appending new windows to the end.

SqueezeTitle [{ *squeeze-list* }]

This variable indicates that **twm** should attempt to use the SHAPE extension to make titlebars occupy only as much screen space as they need, rather than extending all the way across the top of the window. The optional *squeeze-list* may be used to control the location of the squeezed titlebar along the top of the window. It contains entries of the form:

"*name*"   *justification*   *num*   *denom*

where *name* is a window name, *justification* is either **left**, **center**, or **right**, and *num* and *denom* are numbers specifying a ratio giving the relative position about which the titlebar is justified. The ratio is measured from left to right if the numerator is positive, and right to left if negative. A denominator of 0 indicates that the numerator should be measured in pixels. For convenience, the ratio 0/0 is the same as 1/2 for **center** and -1/1 for **right**. For example:

```
SqueezeTitle
{
        "XTerm"     left        0     0
        "xterm1"    left        1     3
        "xterm2"    left        2     3
        "oclock"center      0     0
        "emacs"     right       0     0
}
```

The **DontSqueezeTitle** list can be used to turn off squeezing on certain titles.

StartIconified [{ *win-list* }]
This variable indicates that client windows should initially be left as icons until explicitly deiconified by the user. If the optional *win-list* is given, only those windows will be started iconic. This is useful for programs that do not support an **-iconic** command line option or resource.

TitleBackground *string* [{ *win-list* }]
This variable specifies the background color used in titlebars, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. The default is "white".

TitleButtonBorderWidth *pixels*
This variable specifies the width in pixels of the border surrounding titlebuttons. This is typically set to 0 to allow titlebuttons to take up as much space as possible and to not have a border. The default is 1.

TitleFont *string*
This variable specifies the font to used for displaying window names in titlebars. The default is "variable".

TitleForeground *string* [{ *win-list* }]
This variable specifies the foreground color used in titlebars, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. The default is "black".

TitlePadding *pixels*
This variable specifies the distance between the various buttons, text, and highlight areas in the titlebar. The default is 8 pixels.

UnknownIcon *string*
This variable specifies the filename of a bitmap file to be used as the default icon. This bitmap will be used as the icon of all clients which do not provide an icon bitmap and are not listed in the **Icons** list.

UsePPosition *string*
This variable specifies whether or not **twm** should honor program-requested locations (given by the **PPosition** flag in the WM_NORMAL_HINTS property)

in the absence of a user-specified position. The argument *string* may have one of three values: "**off**" (the default) indicating that **twm** should ignore the program-supplied position, "**on**" indicating that the position should be used, and "**non-zero**" indicating that the position should used if it is other than (0,0). The latter option is for working around a bug in older toolkits.

WarpCursor [{ *win-list* }]
This variable indicates that the pointer should be warped into windows when they are deiconified. If the optional *win-list* is given, the pointer will only be warped when those windows are deiconified.

WindowRing { *win-list* }
This variable specifies a list of windows along which the **f.warpring** function cycles.

WarpUnmapped
This variable indicates that that the **f.warpto** function should deiconify any iconified windows it encounters. This is typically used to make a key binding that will pop a particular window (such as **xmh**), no matter where it is. The default is for **f.warpto** to ignore iconified windows.

XorValue *number*
This variable specifies the value to use when drawing window outlines for moving and resizing. This should be set to a value that will result in a variety of of distinguishable colors when exclusive-or'ed with the contents of the user's typical screen. Setting this variable to 1 often gives nice results if adjacent colors in the default colormap are distinct. By default, **twm** will attempt to cause temporary lines to appear at the opposite end of the colormap from the graphics.

Zoom [*count*]
This variable indicates that outlines suggesting movement of a window to and from its iconified state should be displayed whenever a window is iconified or deiconified. The optional *count* argument specifies the number of outlines to be drawn. The default count is 8.

The following variables must be set after the fonts have been assigned, so it is usually best to put them at the end of the variables or beginning of the bindings sections:

DefaultFunction *function*
This variable specifies the function to be executed when a key or button event is received for which no binding is provided. This is typically bound to **f.nop**, **f.beep**, or a menu containing window operations.

WindowFunction *function*
This variable specifies the function to execute when a window is selected from the **TwmWindows** menu. If this variable is not set, the window will be deiconified and raised.

BINDINGS

After the desired variables have been set, functions may be attached titlebuttons and key and pointer buttons. Titlebuttons may be added from the left or right side and appear in the titlebar from left-to-right according to the order in which they are specified. Key and pointer button bindings may be given in any order.

Titlebuttons specifications must include the name of the pixmap to use in the button box and the function to be invoked when a pointer button is pressed within them:

LeftTitleButton "*bitmapname*" = *function*

or

RightTitleButton "*bitmapname*" = *function*

The *bitmapname* may refer to one of the built-in bitmaps (which are scaled to match **TitleFont**) by using the appropriate colon-prefixed name described above.

Key and pointer button specifications must give the modifiers that must be pressed, over which parts of the screen the pointer must be, and what function is to be invoked. Keys are given as strings containing the appropriate keysym name; buttons are given as the keywords **Button1-Button5**:

```
"FP1"          = modlist : context : function
Button1        = modlist : context : function
```

The **modlist** is any combination of the modifier names **shift**, **control**, **lock**, **meta**, **mod1**, **mod2**, **mod3**, **mod4**, or **mod5** (which may be abbreviated as **s**, **c**, **l**, **m**, **m1**, **m2**, **m3**, **m4**, **m5**, respectively) separated by a vertical bar (|). Similarly, the *context* is any combination of **window**, **title**, **icon**, **root**, **frame**, **iconmgr**, their first letters (**iconmgr** abbreviation is **m**), or **all**, separated by a vertical bar. The *function* is any of the **f.** keywords described below. For example, the default startup file contains the following bindings:

```
Button1=      : root          : f.menu "TwmWindows"
Button1= m    : window | icon: f.function "move-or-lower"
Button2= m    : window | icon: f.iconify
Button3= m    : window | icon: f.function "move-or-raise"
Button1=      : title         : f.function "move-or-raise"
Button2=      : title         : f.raiselower
Button1=      : icon          : f.function "move-or-iconify"
Button2=      : icon          : f.iconify
Button1=      : iconmgr       : f.iconify
Button2=      : iconmgr       : f.iconify
```

A user who wanted to be able to manipulate windows from the keyboard could use the following bindings:

```
"F1"      =     : all          : f.iconify
"F2"      =     : all          : f.raiselower
"F3"      =     : all          : f.warpring "next"
"F4"      =     : all          : f.warpto "xmh"
"F5"      =     : all          : f.warpto "emacs"
"F6"      =     : all          : f.colormap "next"
"F7"      =     : all          : f.colormap "default"
"F20"     =     : all          : f.warptoscreen "next"
```

```
"Left"        = m     : all         : f.backiconmgr
"Right"       = m | s : all         : f.forwiconmgr
"Up"          = m     : all         : f.upiconmgr
"Down"        = m | s : all         : f.downiconmgr
```

twm provides many more window manipulation primitives than can be conveniently stored in a titlebar, menu, or set of key bindings. Although a small set of defaults are supplied (unless the **NoDefaults** is specified), most users will want to have their most common operations bound to key and button strokes. To do this, **twm** associates names with each of the primitives and provides *user-defined functions* for building higher level primitives and *menus* for interactively selecting among groups of functions.

User-defined functions contain the name by which they are referenced in calls to **f.function** and a list of other functions to execute. For example:

```
Function "move-or-lower"      { f.move f.deltastop f.lower }
Function "move-or-raise"      { f.move f.deltastop f.raise }
Function "move-or-iconify"    { f.move f.deltastop f.iconify }
Function "restore-colormap"   { f.colormap "default" f.lower }
```

The function name must be used in **f.function** exactly as it appears in the function specification.

In the descriptions below, if the function is said to operate on the selected window, but is invoked from a root menu, the cursor will be changed to the **Select** cursor and the next window to receive a button press will be chosen:

! *string* This is an abbreviation for **f.exec** *string.*

f.autoraise
This function toggles whether or not the selected window is raised whenever entered by the pointer. See the description of the variable **AutoRaise**.

f.backiconmgr
This function warps the pointer to the previous column in the current icon manager, wrapping back to the previous row if necessary.

f.beep This function sounds the keyboard bell.

f.bottomzoom
This function is similar to the **f.fullzoom** function, but resizes the window to fill only the bottom half of the screen.

f.circledown
This function lowers the top-most window that occludes another window.

f.circleup
This function raises the bottom-most window that is occluded by another window.

f.colormap *string*
This function rotates the colormaps (obtained from the WM_COLORMAP_WINDOWS property on the window) that **twm** will display when the pointer is in this window. The argument *string* may have one of the

following values: "**next**", "**prev**", and "**default**". It should be noted here that in general, the installed colormap is determined by keyboard focus. A pointer driven keyboard focus will install a private colormap upon entry of the window owning the colormap. Using the click to type model, private colormaps will not be installed until the user presses a mouse button on the target window.

f.deiconify
This function deiconifies the selected window. If the window is not an icon, this function does nothing.

f.delete This function sends the WM_DELETE_WINDOW message to the selected window if the client application has requested it through the WM_PROTOCOLS window property. The application is supposed to respond to the message by removing the indicated window. If the window has not requested WM_DELETE_WINDOW messages, the keyboard bell will be rung indicating that the user should choose an alternative method. Note this is very different from f.destroy. The intent here is to delete a single window, not necessarily the entire application.

f.deltastop
This function allows a user-defined function to be aborted if the pointer has been moved more than *MoveDelta* pixels. See the example definition given for **Function** "**move-or-raise**" at the beginning of the section.

f.destroy
This function instructs the X server to close the display connection of the client that created the selected window. This should only be used as a last resort for shutting down runaway clients. See also f.delete.

f.downiconmgr
This function warps the pointer to the next row in the current icon manger, wrapping to the beginning of the next column if necessary.

f.exec *string*
This function passes the argument *string* to /bin/sh for execution. In multiscreen mode, if *string* starts a new X client without giving a display argument, the client will appear on the screen from which this function was invoked.

f.focus This function toggles the keyboard focus of the server to the selected window, changing the focus rule from pointer-driven if necessary. If the selected window already was focused, this function executes an **f.unfocus**.

f.forcemove
This function is like **f.move** except that it ignores the **DontMoveOff** variable.

f.forwiconmgr
This function warps the pointer to the next column in the current icon manager, wrapping to the beginning of the next row if necessary.

f.fullzoom
This function resizes the selected window to the full size of the display or else restores the original size if the window was already zoomed.

f.function *string*
This function executes the user-defined function whose name is specified by the argument *string.*

f.hbzoom
This function is a synonym for **f.bottomzoom**.

f.hideiconmgr
This function unmaps the current icon manager.

f.horizoom
This variable is similar to the **f.zoom** function except that the selected window is resized to the full width of the display.

f.htzoom
This function is a synonym for **f.topzoom**.

f.hzoom
This function is a synonym for **f.horizoom**.

f.iconify
This function iconifies or deiconifies the selected window or icon, respectively.

f.identify
This function displays a summary of the name and geometry of the selected window. Clicking the pointer or pressing a key in the window will dismiss it.

f.lefticonmgr
This function similar to **f.backiconmgr** except that wrapping does not change rows.

f.leftzoom
This variable is similar to the **f.bottomzoom** function but causes the selected window is only resized to the left half of the display.

f.lower This function lowers the selected window.

f.menu *string*
This function invokes the menu specified by the argument *string.* Cascaded menus may be built by nesting calls to **f.menu**.

f.move This function drags an outline of the selected window (or the window itself if the **OpaqueMove** variable is set) until the invoking pointer button is released. Double clicking within the number of milliseconds given by **ConstrainedMove-Time** warps the pointer to the center of the window and constrains the move to be either horizontal or vertical depending on which grid line is crossed. To abort a move, press another button before releasing the first button.

f.nexticonmgr
This function warps the pointer to the next icon manager containing any windows on the current or any succeeding screen.

f.nop This function does nothing and is typically used with the **DefaultFunction** or **WindowFunction** variables or to introduce blank lines in menus.

f.previconmgr

This function warps the pointer to the previous icon manager containing any windows on the current or preceding screens.

f.quit This function causes **twm** to restore the window's borders and exit. If **twm** is the first client invoked from **xdm**, this will result in a server reset.

f.raise This function raises the selected window.

f.raiselower
This function raises the selected window to the top of the stacking order if it is occluded by any windows, otherwise the window will be lowered.

f.refresh
This function causes all windows to be refreshed.

f.resize This function displays an outline of the selected window. Crossing a border (or setting **AutoRelativeResize**) will cause the outline to begin to rubber band until the invoking button is released. To abort a resize, press another button before releasing the first button.

f.restart This function kills and restarts **twm**.

f.righticonmgr
This function is similar to **f.nexticonmgr** except that wrapping does not change rows.

f.rightzoom
This variable is similar to the **f.bottomzoom** function except that the selected window is only resized to the right half of the display.

f.saveyourself
This function sends a WM_SAVEYOURSELF message to the selected window if it has requested the message in its WM_PROTOCOLS window property. Clients that accept this message are supposed to checkpoint all state associated with the window and update the WM_COMMAND property as specified in the ICCCM. If the selected window has not selected for this message, the keyboard bell will be rung.

f.showiconmgr
This function maps the current icon manager.

f.sorticonmgr
This function sorts the entries in the current icon manager alphabetically. See the variable **SortIconManager**.

f.title This function provides a centered, unselectable item in a menu definition. It should not be used in any other context.

f.topzoom
This variable is similar to the **f.bottomzoom** function except that the selected window is only resized to the top half of the display.

f.unfocus
This function resets the focus back to pointer-driven. This should be used when a focused window is no longer desired.

f.upiconmgr
This function warps the pointer to the previous row in the current icon manager, wrapping to the last row in the same column if necessary.

f.vlzoom
This function is a synonym for **f.leftzoom**.

f.vrzoom
This function is a synonym for **f.rightzoom**.

f.warpring *string*
This function warps the pointer to the next or previous window (as indicated by the argument **string**, which may be "**next**" or **"prev"**) specified in the **WindowR-ing** variable.

f.warpto *string*
This function warps the pointer to the window which has a name or class that matches *string.* If the window is iconified, it will be deiconified if the variable **WarpUnmapped** is set or else ignored.

f.warptoiconmgr *string*
This function warps the pointer to the icon manager entry associated with the window containing the pointer in the icon manager specified by the argument *string.* If *string* is empty (i.e. ""), the current icon manager is chosen.

f.warptoscreen *string*
This function warps the pointer to the screen specified by the argument *string.* **String** may be a number (e.g. "**0**" or "**1**"), the word **"next"** (indicating the current screen plus 1, skipping over any unmanaged screens), the word "**back**" (indicat-ing the current screen minus 1, skipping over any unmanaged screens), or the word "**prev**" (indicating the last screen visited.

f.winrefresh
This function is similar to the **f.refresh** function except that only the selected window is refreshed.

f.zoom This function is similar to the **f.fullzoom** function, except that the only the height of the selected window is changed.

MENUS

Functions may be grouped and interactively selected using pop-up (when bound to a pointer button) or pull-down (when associated with a titlebutton) menus. Each menu specification contains the name of the menu as it will be referred to by **f.menu**, optional default foreground and background colors, the list of item names and the functions they should invoke, and optional foreground and background colors for individual items:

```
Menu "menuname" [ ("deffore":"defback") ]
{
        string1   [ ("fore1":"backn")]       function1
        string2   [ ("fore2":"backn")]       function2
                  .
                  .
                  .
```

```
                stringN  [ ("foreN":"backN")]         functionN
        }
```

The *menuname* is case-sensitive. The optional *deffore* and *defback* arguments specify the foreground and background colors used on a color display to highlight menu entries. The *string* portion of each menu entry will be the text which will appear in the menu. The optional *fore* and *back* arguments specify the foreground and background colors of the menu entry when the pointer is not in the entry. These colors will only be used on a color display. The default is to use the colors specified by the **MenuForeground** and **Menu-***Background* variables. The *function* portion of the menu entry is one of the functions, including any user-defined functions, or additional menus.

There is a special menu named **TwmWindows** which contains the names of all of the client and **twm**-supplied windows. Selecting an entry will cause the **WindowFunction** to be executed on that window. If **WindowFunction** hasn't been set, the window will be deiconified and raised.

ICONS

twm supports several different ways of manipulating iconified windows. The common pixmap-and-text style may be laid out by hand or automatically arranged as described by the **IconRegion** variable. In addition, a terse grid of icon names, called an icon manager, provides a more efficient use of screen space as well as the ability to navigate among windows from the keyboard.

An icon manager is a window that contains names of selected or all windows currently on the display. In addition to the window name, a small button using the default iconify symbol will be displayed to the left of the name when the window is iconified. By default, clicking on an entry in the icon manager performs **f.iconify**. To change the actions taken in the icon manager, use the the **iconmgr** context when specifying button and keyboard bindings.

Moving the pointer into the icon manager also directs keyboard focus to the indicated window (setting the focus explicitly or else sending synthetic events **NoTitleFocus** is set). Using the **f.upiconmgr**, **f.downiconmgr f.lefticonmgr**, and **f.righticonmgr** functions, the input focus can be changed between windows directly from the keyboard.

BUGS

The resource manager should have been used instead of all of the window lists.

The **IconRegion** variable should take a list.

Double clicking very fast to get the constrained move function will sometimes cause the window to move, even though the pointer is not moved.

If **IconifyByUnmapping** is on and windows are listed in **IconManagerDontShow** but not in **DontIconifyByUnmapping**, they may be lost if they are iconified and no bindings to **f.menu "TwmWindows"** or **f.warpto** are setup.

FILES

$HOME/.twmrc.*<screen number>*
$HOME/.twmrc
/usr/openwin/lib/X11/system.twmrc

ENVIRONMENT VARIABLES

DISPLAY
This variable is used to determine which X server to use. It is also set during **f.exec** so that programs come up on the proper screen.

HOME This variable is used as the prefix for files that begin with a tilde and for locating the **twm** startup file.

SEE ALSO

X11(7), **xdm**(1), **xrdb**(1)

COPYRIGHT

Portions copyright 1988 Evans & Sutherland Computer Corporation; portions copyright 1989 Hewlett-Packard Company and the Massachusetts Institute of Technology, See **X11**(7) for a full statement of rights and permissions.

AUTHORS

Tom LaStrange, Solbourne Computer; Jim Fulton, MIT X Consortium; Steve Pitschke, Stardent Computer; Keith Packard, MIT X Consortium; Dave Sternlicht, MIT X Consortium; Dave Payne, Apple Computer.

NAME

viewres – graphical class browser for Xt

SYNOPSIS

viewres [-option ...]

DESCRIPTION

The **viewres** program displays a tree showing the widget class hierarchy of the Athena Widget Set. Each node in the tree can be expanded to show the resources that the corresponding class adds (i.e. does not inherit from its parent) when a widget is created. This application allows the user to visually examine the structure and inherited resources for the Athena Widget Set.

OPTIONS

Viewres accepts all of the standard toolkit command line options as well as the following:

–**top** *name*
: This option specifies the name of the highest widget in the hierarchy to display. This is typically used to limit the display to a subset of the tree. The default is *Object*.

–**variable**
: This option indicates that the widget variable names (as declared in header files) should be displayed in the nodes rather than the widget class name. This is sometimes useful to distinguish widget classes that share the same name (such as *Text*).

–**vertical**
: This option indicates that the tree should be displayed top to bottom rather left to right.

VIEW MENU

The way in which the tree is displayed may be changed with through the entries in the **View** menu:

Show Variable Names
: This entry causes the node labels to be set to the variable names used to declare the corresponding widget class. This operation may also be performed with the **SetLabelType(variable)** translation.

Show Class Names
: This entry causes the node labels to be set to the class names used when specifying resources. This operation may also be performed with the **SetLabelType(class)** translation.

Layout Horizontal
: This entry causes the tree to be laid out from left to right. This operation may also be performed with the *SetOrientation(West)* translation.

Layout Vertical
: This entry causes the tree to be laid out from top to bottom. This operation may also be performed with the *SetOrientation(North)* translation.

Show Resource Boxes
: This entry expands the selected nodes (see next section) to show the new widget

and constraint resources. This operation may also be performed with the *Resources(on)* translation.

Hide Resource Boxes
This entry removes the resource displays from the selected nodes (usually to conserve space). This operation may also be performed with the *Resources(off)* translation.

SELECT MENU

Resources for a single widget class can be displayed by clicking **Button2** on the corresponding node, or by adding the node to the selection list with **Button1** and using the **Show Resource Boxes** entry in the **View** menu. Since **Button1** actually toggles the selection state of a node, clicking on a selected node will cause it to be removed from the selected list.

Collections of nodes may also be selected through the various entries in the **Select** menu:

Unselect All
This entry removes all nodes from the selection list. This operation may also be performed with the *Select(nothing)* translation.

Select All
This entry adds all nodes to the selection list. This operation may also be performed with the *Select(all)* translation.

Invert All
This entry adds unselected nodes to, and removes selected nodes from, the selection list. This operation may also be performed with the *Select(invert)* translation.

Select Parent
This entry selects the immediate parents of all selected nodes. This operation may also be performed with the *Select(parent)* translation.

Select Ancestors
This entry recursively selects all parents of all selected nodes. This operation may also be performed with the *Select(ancestors)* translation.

Select Children
This entry selects the immediate children of all selected nodes. This operation may also be performed with the *Select(children)* translation.

Select Descendants
This entry recursively selects all children of all selected nodes. This operation may also be performed with the *Select(descendants)* translation.

Select Has Resources
This entry selects all nodes that add new resources (regular or constraint) to their corresponding widget classes. This operation may also be performed with the *Select(resources)* translation.

Select Shown Resource Boxes
This entry selects all nodes whose resource boxes are currently expanded (usually so that they can be closed with **Hide Resource Boxes**). This operation may

also be performed with the *Select(shown)* translation.

ACTIONS

The following application actions are provided:

Quit()
: This action causes **viewres** to exit.

SetLabelType(*type***)**
: This action sets the node labels to display the widget *variable* or *class* names, according to the argument *type*.

SetOrientation(*direction***)**
: This action sets the root of the tree to be one of the following areas of the window: *West*, *North*, *East*, or *South*.

Select(*what***)**
: This action selects the indicated nodes, as described in the **VIEW MENU** section: *nothing* (unselects all nodes), *invert*, *parent*, *ancestors*, *children*, *descendants*, *resources*, *shown*.

Resources(*op***)**
: This action turns *on*, *off*, or *toggles* the resource boxes for the selected nodes. If invoked from within one of the nodes (through the keyboard or pointer), only that node is used.

WIDGET HIERARCHY

Resources may be specified for the following widgets:

```
Viewres viewres
        Paned pane
                Box buttonbox
                        Command quit
                        MenuButton view
                                SimpleMenu viewMenu
                                        SmeBSB layoutHorizontal
                                        SmeBSB layoutVertical
                                        SmeLine line1
                                        SmeBSB namesVariable
                                        SmeBSB namesClass
                                        SmeLine line2
                                        SmeBSB viewResources
                                        SmeBSB viewNoResources
                        MenuButton select
                                SimpleMenu selectMenu
                                        SmeBSB unselect
                                        SmeBSB selectAll
                                        SmeBSB selectInvert
                                        SmeLine line1
                                        SmeBSB selectParent
                                        SmeBSB selectAncestors
```

```
                                          SmeBSB selectChildren
                                          SmeBSB selectDescendants
                                          SmeLine line2
                                          SmeBSB selectHasResources
                                          SmeBSB selectShownResources
                  Form treeform
                        Porthole porthole
                              Tree tree
                                    Box variable-name
                                          Toggle variable-name
                                          List variable-name
                        Panner panner
```

where *variable-name* is the widget variable name of each node.

SEE ALSO

X11(7), **xrdb**(1), **listres**(1), **editres**(1), **appres**(1), appropriate widget documents

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

winsysck – check which window system protocols are available

SYNOPSIS

winsysck [–**va**] [–**display** *displaystring*] *protocol* [. . .]

DESCRIPTION

The **winsysck** command determines if the specified window system *protocol* is available to the user. By default, the **winsysck** command exits as soon as the first available *protocol* is found, although this behaviour can be modified by the **-a** option (see below).

OPTIONS

The following options can be used to modify the behaviour of the **winsysck** command.

–**a** Continue to check the availability of the specified *protocol*s even after determining that one or more previously specified *protocol*s are available. This is not particularly useful without the –**v** option (see below).

–**v** Print the name of the first available *protocol* on the standard output. When combined with the –**a** option (see above), print the name of all available *protocol*s on the standard output, separated by newlines.

–**display** *displaystring*
Use the display *displaystring* when trying to determine if the **x11** *protocol* is available.

KNOWN PROTOCOLS

The following are known values for *protocol*.

x11 Determines if a connection can be made using the X11 Window System protocol.

news Determines if a connection can be made using the Network-extensible Window System protocol.

x11news
Determines if connections can be made to an X11 / NeWS server. In addition to being able to establish connections using both the X11 Window System and Network-extensible Window System protocols, this requires that these two connections actually interact with the same window server.

sunview
Determines if connections can be made using the SunView Window System protocol.

EXAMPLES

Determine if an X11 Window System connection can be made:

```
example% if winsysck x11 ; then ...
```

Determine if an X11 Window System connection can be made to the display “displayhost:0.0”:

```
example% if winsysck -display displayhost:0.0 x11 ; then ...
```

Print the known protocols which are available:

```
example% winsysck -v -a x11 news x11news sunview
```

Print which window system should be used, given the preference for NeWS over SunView, and X11 / NeWS over NeWS:

```
example% winsysck -v x11news news sunview
```

SEE ALSO

Xsun(1)

LIMITATIONS

There is no way to determine whether a SunView connection is actually connected to an X11 / NeWS server with SunView compatability enabled or to a SunView server.

BUGS

None known.

DIAGNOSTICS

Exit status is 0 if any *protocol*s are available, 1 if none are available, 2 for usage errors.

NAME

worm – draw wiggly worms

SYNOPSIS

worm [**–l** *length*] [**–s** *size*] [**–n** *number*] [**–d** *connection*] [**–g** *geometry*] [**–R**] [**–C**] [**–S**]

DESCRIPTION

worm draws wiggly worms. It is adapted from a concept in the December 1987 issue of Scientific American. Playing with the various parameters can create strange effects. Pressing any key in the worm window will cause them to freeze; pressing again will thaw.

OPTIONS

–S Screensaver. Takes over entire screen.

–C Chromocolor. Worms change colors as they crawl.

–R Rotate colormap. The colormap constantly changes.

–n *number*
Make *number* worms. Default is 50.

–l *length*
Worms are if length *length.* A negative value means infinite length.

–size *size*
Worms are *size* pixels wide.

–display *connection*
Connect to X server display, *connection.*

–geometry *geomspec*
Create window using *geomspec.*

SEE ALSO

X11(7)

COPYRIGHT

NAME

X11 – a portable, network-transparent window system

SYNOPSIS

There is no X11 command per se. This manual page is adapted from the X manual page supplied with the MIT sample server and is included with OpenWindows for reference.

The X Window System is a network transparent window system developed at MIT which runs on a wide range of computing and graphics machines.

The X Consortium requests that the following names be used when referring to this software:

X
X Window System
X Version 11
X Window System, Version 11
X11

DESCRIPTION

X Window System servers run on computers with bitmap displays. The server distributes user input to and accepts output requests from various client programs through a variety of different interprocess communication channels. Although the most common case is for the client programs to be running on the same machine as the server, clients can be run transparently from other machines (including machines with different architectures and operating systems) as well.

X supports overlapping hierarchical subwindows and text and graphics operations, on both monochrome and color displays. For a full explanation of the functions that are available, see the **Xlib - C Language X Interface** manual, the *X Window System Protocol* specification, the *X Toolkit Intrinsics - C Language Interface* manual, and various toolkit documents.

Many utilities, window managers, games, toolkits, etc. are available from the user-contributed software. See your site administrator for details.

STARTING UP

See **openwin**(1) for information on starting the server and an initial set of client applications.

DISPLAY NAMES

From the user’s prospective, every X server has a *display name* of the form:

hostname:displaynumber.screennumber

This information is used by the application to determine how it should connect to the server and which screen it should use by default (on displays with multiple monitors):

hostname
: The *hostname* specifies the name of the machine to which the display is physically connected. If the hostname is not given, the most efficient way of communicating to a server on the same machine will be used.

displaynumber

The phrase "display" is usually used to refer to collection of monitors that share a common keyboard and pointer (mouse, tablet, etc.). Most workstations tend to only have one keyboard, and therefore, only one display. Larger, multi-user systems, however, will frequently have several displays so that more than one person can be doing graphics work at once. To avoid confusion, each display on a machine is assigned a *display number* (beginning at 0) when the X server for that display is started. The display number must always be given in a display name.

screennumber

Some displays share a single keyboard and pointer among two or more monitors. Since each monitor has its own set of windows, each screen is assigned a *screen number* (beginning at 0) when the X server for that display is started. If the screen number is not given, then screen 0 will be used.

On POSIX systems, the default display name is stored in your DISPLAY environment variable. This variable is set automatically by the **xterm**(1) terminal emulator. However, when you log into another machine on a network, you'll need to set DISPLAY by hand to point to your display. Examples include,

```
% setenv DISPLAY myws:0

$ DISPLAY=myws:0; export DISPLAY
```

Finally, most X programs accept a command line option of **-display** *displayname* to temporarily override the contents of DISPLAY. This is most commonly used to pop windows on another person's screen or as part of a "remote shell" command to start an xterm pointing back to your display. For example,

```
% xeyes -display joesws:0 -geometry 1000x1000+0+0
% rsh big xterm -display myws:0 -ls </dev/null &
```

X servers listen for connections on a variety of different communications channels (network byte streams, shared memory, etc.). Since there can be more than one way of contacting a given server, The *hostname* part of the display name is used to determine the type of channel (also called a transport layer) to be used. The sample servers from MIT support the following types of connections:

local

The hostname part of the display name should be the empty string. For example: *:0*, *:1*, and *:0.1*. The most efficient local transport will be chosen.

TCP/IP

The hostname part of the display name should be the server machine's IP address name. Full Internet names, abbreviated names, and IP addresses are all allowed. For example: *expo.lcs.mit.edu:0*, *expo:0*, *18.30.0.212:0*, *bigmachine:1*, and *hydra:0.1*.

DECnet

The hostname part of the display name should be the server machine's nodename followed by two colons instead of one. For example: *myws::0, big::1*, and *hydra::0.1*. Note that DECnet connections are not supported under Solaris x86 or PowerPC.

ACCESS CONTROL

The sample server provides two types of access control: an authorization protocol which provides a list of ''magic cookies'' clients can send to request access, and a list of hosts from which connections are always accepted. **Xdm** initializes magic cookies in the server, and also places them in a file accessible to the user. Normally, the list of hosts from which connections are always accepted should be empty, so that only clients with are explicitly authorized can connect to the display. When you add entries to the host list (with **xhost**(1)), the server no longer performs any authorization on connections from those machines.

The file for authorization which both **xdm**(1) and **Xlib** use can be specified with the environment variable **XAUTHORITY**, and defaults to the file **.Xauthority** in the home directory. **Xdm** uses **$HOME/.Xauthority** and will create it or merge in authorization records if it already exists when a user logs in.

To manage a collection of authorization files containing a collection of authorization records use **xauth**(1). This program allows you to extract records and insert them into other files. Using this, you can send authorization to remote machines when you login. As the files are machine-independent, you can also simply copy the files or use NFS to share them. If you use several machines, and share a common home directory with NFS, then you never really have to worry about authorization files, the system should work correctly by default. Note that magic cookies transmitted ''in the clear'' over NFS or using **ftp**(1) or **rpc**(3N) can be ''stolen'' by a network eavesdropper, and as such may enable unauthorized access. In many environments this level of security is not a concern, but if it is, you need to know the exact semantics of the particular magic cookie to know if this is actually a problem.

GEOMETRY SPECIFICATIONS

One of the advantages of using window systems instead of hardwired terminals is that applications don't have to be restricted to a particular size or location on the screen. Although the layout of windows on a display is controlled by the window manager that the user is running (described below), most X programs accept a command line argument of the form **-geometry** *WIDTHxHEIGHT+XOFF+YOFF* (where *WIDTH*, *HEIGHT*, *XOFF*, and *YOFF* are numbers) for specifying a preferred size and location for this application's main window.

The *WIDTH* and *HEIGHT* parts of the geometry specification are usually measured in either pixels or characters, depending on the application. The *XOFF* and *YOFF* parts are measured in pixels and are used to specify the distance of the window from the left or right and top and bottom edges of the screen, respectively. Both types of offsets are measured from the indicated edge of the screen to the corresponding edge of the window. The X offset may be specified in the following ways:

+XOFF The left edge of the window is to be placed *XOFF* pixels in from the left edge of the screen (i.e. the X coordinate of the window's origin will be *XOFF*). *XOFF* may be negative, in which case the window's left edge will be off the screen.

-*XOFF* The right edge of the window is to be placed *XOFF* pixels in from the right edge of the screen. *XOFF* may be negative, in which case the window's right edge will be off the screen.

The Y offset has similar meanings:

+*YOFF* The top edge of the window is to be *YOFF* pixels below the top edge of the screen (i.e. the Y coordinate of the window's origin will be *YOFF*). *YOFF* may be negative, in which case the window's top edge will be off the screen.

-*YOFF* The bottom edge of the window is to be *YOFF* pixels above the bottom edge of the screen. *YOFF* may be negative, in which case the window's bottom edge will be off the screen.

Offsets must be given as pairs; in other words, in order to specify either *XOFF* or *YOFF* both must be present. Windows can be placed in the four corners of the screen using the following specifications:

+0+0	upper left hand corner.
-0+0	upper right hand corner.
-0-0	lower right hand corner.
+0-0	lower left hand corner.

In the following examples, a terminal emulator will be placed in roughly the center of the screen and a load average monitor, mailbox, and clock will be placed in the upper right hand corner:

```
xterm -fn 6x10 -geometry 80x24+30+200 &
xclock -geometry 48x48-0+0 &
xload -geometry 48x48-96+0 &
xbiff -geometry 48x48-48+0 &
```

WINDOW MANAGERS

The layout of windows on the screen is controlled by special programs called *window managers*. Although many window managers will honor geometry specifications as given, others may choose to ignore them (requiring the user to explicitly draw the window's region on the screen with the pointer, for example).

Since window managers are regular (albeit complex) client programs, a variety of different user interfaces can be built. OpenWindows comes with a window manager named **olwm**(1) which supports overlapping windows, popup menus, point-and-click or click-to-type input models, title bars, nice icons, and many other features.

FONT NAMES

Collections of characters for displaying text and symbols in X are known as *fonts*. A font typically contains images that share a common appearance and look nice together (for example, a single size, boldness, slant, and character set).

The list of font directories in which the server looks when trying to find a font is controlled by the *font path*. Although most installations will choose to have the server start up with all of the commonly used font directories, the font path can be changed at any time with the **xset**(1) program. However, it is important to remember that the directory names are on the **server**'s machine, not on the application's.

The default font path for the OpenWindows server is **/usr/openwin/lib/fonts**.

Font databases are created by running the **mkfontdir**(1) program in the directory containing the compiled versions of the fonts. Whenever fonts are added to a directory, **mkfontdir** should be rerun so that the server can find the new fonts. To make the server reread the font database, reset the font path with the **xset** program. For example, to add a font to a private directory, the following commands could be used:

```
    % cp newfont.fb ˜/ myfonts
    % mkfontdir ˜/ myfonts
    % xset fp rehash
```

The **xlsfonts**(1) program can be used to list all of the fonts that are found in font databases in the current font path. Font names tend to be fairly long as they contain all of the information needed to uniquely identify individual fonts. However, the sample server supports wildcarding of font names, so the full specification

-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1

could be abbreviated as:

--courier-medium-r-normal--*-100-*-*-*-*-*-**

Because the shell also has special meanings for ∗ and ?, wildcarded font names should be quoted:

```
    % xlsfonts -fn '-*-courier-medium-r-normal--*-100-*-*-*-*-*-*'
```

If more than one font in a given directory in the font path matches a wildcarded font name, the choice of which particular font to return is left to the server. However, if fonts from more than one directory match a name, the returned font will always be from the first such directory in the font path.

COLOR NAMES

Most applications provide ways of tailoring (usually through resources or command line arguments) the colors of various elements in the text and graphics they display. Although black and white displays don't provide much of a choice, color displays frequently allow anywhere between 16 and 16 million different colors.

Colors are usually specified by their commonly-used names (for example, *red*, *white*, or *medium slate blue*). The server translates these names into appropriate screen colors using a color database that can usually be found in **/usr/openwin/lib/rgb.txt .** Color names are case-insensitive, meaning that *red*, *Red*, and *RED* all refer to the same color.

Many applications also accept color specifications of the following form:

```
#rgb
#rrggbb
```

#rrrgggbbb
#rrrrggggbbbb

where *r*, *g*, and *b* are hexadecimal numbers indicating how much *red*, *green*, and *blue* should be displayed (zero being none and ffff being on full). Each field in the specification must have the same number of digits (e.g., #rrgb or #gbb are not allowed). Fields that have fewer than four digits (e.g. #rgb) are padded out with zero's following each digit (e.g. #r000g000b000). The eight primary colors can be represented as:

black	#000000000000 (no color at all)
red	#ffff00000000
green	#0000ffff0000
blue	#00000000ffff
yellow	#ffffffff0000 (full red and green, no blue)
magenta	#ffff0000ffff
cyan	#0000ffffffff
white	#ffffffffffff (full red, green, and blue)

Unfortunately, RGB color specifications are highly unportable since different monitors produce different shades when given the same inputs. Similarly, color names aren't portable because there is no standard naming scheme and because the color database needs to be tuned for each monitor.

Application developers should take care to make their colors tailorable.

KEYS

The X keyboard model is broken into two layers: server-specific codes (called *keycodes*) which represent the physical keys, and server-independent symbols (called *keysyms*) which represent the letters or words that appear on the keys. Two tables are kept in the server for converting keycodes to keysyms:

modifier list
Some keys (such as Shift, Control, and Caps Lock) are known as *modifier* and are used to select different symbols that are attached to a single key (such as Shift-a generates a capital A, and Control-l generates a formfeed character ^L). The server keeps a list of keycodes corresponding to the various modifier keys. Whenever a key is pressed or released, the server generates an *event* that contains the keycode of the indicated key as well as a mask that specifies which of the modifier keys are currently pressed. Most servers set up this list to initially contain the various shift, control, and shift lock keys on the keyboard.

keymap table
Applications translate event keycodes and modifier masks into keysyms using a *keysym table* which contains one row for each keycode and one column for various modifier states. This table is initialized by the server to correspond to normal typewriter conventions, but is only used by client programs.

Although most programs deal with keysyms directly (such as those written with the X Toolkit Intrinsics), most programming libraries provide routines for converting keysyms into the appropriate type of string (such as ISO Latin-1).

OPTIONS

Most X programs attempt to use the same names for command line options and arguments. All applications written with the X Toolkit Intrinsics automatically accept the following options:

–display *display*
: This option specifies the name of the X server to use.

–geometry *geometry*
: This option specifies the initial size and location of the window.

–bg *color***,** **–background** *color*
: Either option specifies the color to use for the window background.

–bd *color***,** **–bordercolor** *color*
: Either option specifies the color to use for the window border.

–bw *number***,** **–borderwidth** *number*
: Either option specifies the width in pixels of the window border.

–fg *color***,** **–foreground** *color*
: Either option specifies the color to use for text or graphics.

–fn *font***,** **-font** *font*
: Either option specifies the font to use for displaying text.

–iconic
: This option indicates that the user would prefer that the application's windows initially not be visible as if the windows had be immediately iconified by the user. Window managers may choose not to honor the application's request.

–name
: This option specifies the name under which resources for the application should be found. This option is useful in shell aliases to distinguish between invocations of an application, without resorting to creating links to alter the executable file name.

–rv, **–reverse**
: Either option indicates that the program should simulate reverse video if possible, often by swapping the foreground and background colors. Not all programs honor this or implement it correctly. It is usually only used on monochrome displays.

+rv
: This option indicates that the program should not simulate reverse video. This is used to override any defaults since reverse video doesn't always work properly.

–selectionTimeout
: This option specifies the timeout in milliseconds within which two communicating applications must respond to one another for a selection request.

–synchronous
: This option indicates that requests to the X server should be sent synchronously, instead of asynchronously. Since **Xlib** normally buffers requests to the server, errors do not necessarily get reported immediately after they occur. This option

turns off the buffering so that the application can be debugged. It should never be used with a working program.

–title *string*
: This option specifies the title to be used for this window. This information is sometimes used by a window manager to provide some sort of header identifying the window.

–xnllanguage *language***[***_territory***][***.codeset***]**
: This option specifies the language, territory, and codeset for use in resolving resource and other filenames.

–xrm *resourcestring*
: This option specifies a resource name and value to override any defaults. It is also very useful for setting resources that don't have explicit command line arguments.

RESOURCES

To make the tailoring of applications to personal preferences easier, X supports several mechanisms for storing default values for program resources (e.g. background color, window title, etc.) Resources are specified as strings of the form

*appname*subname*subsubname...: value*

that are read in from various places when an application is run. By convention, the application name is the same as the program name, but with the first letter capitalized (e.g. *Bitmap* or *Emacs*) although some programs that begin with the letter ‘‘x’’ also capitalize the second letter for historical reasons. The precise syntax for resources is:

```
ResourceLine        =    Comment | ResourceSpec
Comment             =    "!" string | <empty line>
ResourceSpec        =    WhiteSpace ResourceName WhiteSpace ":" WhiteSpace value
ResourceName        =    [Binding] ComponentName {Binding ComponentName}
Binding             =    "." | "*"
WhiteSpace          =    {" " | "\t"}
ComponentName       =    {"a"–"z" | "A"–"Z" | "0"–"9" | "_" | "-"}
value               =    string
string              =    {<any character not including "\n">}
```

Note that elements enclosed in curly braces ({...}) indicate zero or more occurrences of the enclosed elements

To allow values to contain arbitrary octets, the 4-character sequence \nnn, where n is a digit in the range of "0"–"7", is recognized and replaced with a single byte that contains this sequence interpreted as an octal number. For example, a value containing a NULL byte can be stored by specifying "\000".

The **Xlib** routine *XGetDefault(3X)* and the resource utilities within **Xlib** and the X Toolkit Intrinsics obtain resources from the following sources:

RESOURCE_MANAGER root window property
: Any global resources that should be available to clients on all machines should

be stored in the RESOURCE_MANAGER property on the root window using the *xrdb* program. This is frequently taken care of when the user starts up X through the display manager or *xinit*.

application-specific files
Programs that use the X Toolkit Intrinsics will also look in the directories named by the environment variable XUSERFILESEARCHPATH or the environment variable XAPPLRESDIR, plus directories in a standard place (usually under **/usr/openwin/lib/X11**, but this can be overridden with the XFILESEARCHPATH environment variable) for application-specific resources.

–**xrm** *resourcestring*
Applications that use the X Toolkit Intrinsics can have resources specified from the command line. The *resourcestring* is a single resource name and value as shown above. Note that if the string contains characters interpreted by the shell (e.g., asterisk), they must be quoted. Any number of –**xrm** arguments may be given on the command line.

Program resources are organized into groups called *classes*, so that collections of individual resources (each of which are called *instances*) can be set all at once. By convention, the instance name of a resource begins with a lowercase letter and class name with an upper case letter. Multiple word resources are concatenated with the first letter of the succeeding words capitalized. Applications written with the X Toolkit Intrinsics will have at least the following resources:

background (class **Background)**
This resource specifies the color to use for the window background.

borderWidth (class **BorderWidth)**
This resource specifies the width in pixels of the window border.

borderColor (class **BorderColor)**
This resource specifies the color to use for the window border.

Most applications using the X Toolkit Intrinsics also have the resource **foreground** (class **Foreground**), specifying the color to use for text and graphics within the window.

By combining class and instance specifications, application preferences can be set quickly and easily. Users of color displays will frequently want to set Background and Foreground classes to particular defaults. Specific color instances such as text cursors can then be overridden without having to define all of the related resources. For example,

```
bitmap*Dashed:  off
XTerm*cursorColor:  gold
XTerm*multiScroll:  on
XTerm*jumpScroll:  on
XTerm*reverseWrap:  on
XTerm*curses:  on
XTerm*Font:  6x10
XTerm*scrollBar: on
```

```
XTerm*scrollbar*thickness: 5
XTerm*multiClickTime: 500
XTerm*charClass:  33:48,37:48,45-47:48,64:48
XTerm*cutNewline: off
XTerm*cutToBeginningOf3ine: off
XTerm*titeInhibit:  on
XTerm*ttyModes:  intr ^c erase ^? kill ^u
XLoad*Background: gold
XLoad*Foreground: red
XLoad*highlight: black
XLoad*borderWidth: 0
emacs*Geometry:  80x65-0-0
emacs*Background:  #5b7686
emacs*Foreground:  white
emacs*Cursor:  white
emacs*BorderColor:  white
emacs*Font:  6x10
xmag*geometry: -0-0
xmag*borderColor:  white
```

If these resources were stored in a file called **.Xresources** in your home directory, they could be added to any existing resources in the server with the following command:

```
% xrdb -merge $HOME/.Xresources
```

This is how the openwin startup script merges user-specific defaults into any site-wide defaults. All sites are encouraged to set up convenient ways of automatically loading resources. See the **Xlib** manual section *Using the Resource Manager* for more information.

EXAMPLES

The following is a collection of sample command lines for some of the more frequently used commands. For more information on a particular command, please refer to that command's manual page.

```
% xrdb -load $HOME/.Xresources
% xmodmap -e "keysym BackSpace = Delete"
% mkfontdir /usr/local/lib/otherfonts
% xset fp+ /usr/local/lib/otherfonts
% xmodmap $HOME/.keymap.km
% xsetroot -solid '#888'
% xset b 100 400 c 50 s 1800 r on
% xset q
% olwm
% xmag
% xclock -geometry 48x48-0+0 -bg blue -fg white
% xeyes -geometry 48x48-48+0
% xbiff -update 20
% xlsfonts '*helvetica*'
% xlswins -l
% xwininfo -root
% xdpyinfo -display joesworkstation:0
% xhost -joesworkstation
% xrefresh
% xwd | xwud
% bitmap companylogo.bm 32x32
% xcalc -bg blue -fg magenta
% xterm -geometry 80x66-0-0 -name myxterm $*
```

DIAGNOSTICS

A wide variety of error messages are generated from various programs. Various toolkits are encouraged to provide a common mechanism for locating error text so that applications can be tailored easily. Programs written to interface directly to the **Xlib** C language library are expected to do their own error checking.

The default error handler in **Xlib** (also used by many toolkits) uses standard resources to construct diagnostic messages when errors occur. The defaults for these messages are usually stored in **/usr/openwin/lib/XErrorDB**. If this file is not present, error messages will consist of error codes only.

When the X Toolkit Intrinsics encounter errors converting resource strings to the appropriate internal format, no error messages are usually printed. This is convenient when it is desirable to have one set of resources across a variety of displays (e.g. color vs. monochrome, lots of fonts vs. very few, etc.), although it can pose problems for trying to determine why an application might be failing. This behavior can be overridden by the setting the *StringConversionsWarning* resource.

To force the X Toolkit Intrinsics to always print string conversion error messages, the following resource should be placed at the top of the file that gets loaded onto the RESOURCE_MANAGER property using the **xrdb**(1) program (frequently called *.Xresources* or *.Xres* in the user's home directory):

```
*StringConversionWarnings: on
```

To have conversion messages printed for just a particular application, the appropriate instance name can be placed before the asterisk:

```
xterm*StringConversionWarnings: on
```

SEE ALSO

appres(1), **bdftosnf**(1), **bitmap**(1), **mkfontdir**(1), **makebdf**(1), **imake**(1), **listres**(1), **maze**(6), **mkfontdir**(1), **muncher**(6), **oclock**(1), **olwm**(1), **puzzle**(6), **resize**(1), **showsnf**(1), **twm**(1), **xauth**(1), **xbiff**(1), **xcalc**(1), **xclipboard**(1), **xclock**(1), **xditview**(1), **xdm**(1), **xdpyinfo**(1), **xedit**(1), **xev**(6), **xeyes**(6), **xfd**(1), **xfontsel**(1), **xhost**(1), **xinit**(1), **xkill**(1), **xload**(1), **xlogo**(1), **xlsatoms**(1), **xlsclients**(1), **xlsfonts**(1), **xlswins**(1), **xmag**(1), **xman**(1), **xmh**(1), **xmodmap**(1), **xpr**(1), **xprop**(1), **xrdb**(1), **xrefresh**(1), **xset**(1), **xsetroot**(1), **xstdcmap**(1), **xterm**(1), **xview**(7), **xwd**(1), **xwininfo**(1), **xwud**(1)
Xlib – C Language X Interface, *X Toolkit Intrinsics - C Language Interface*, and *Using and*

Specifying X Resources

COPYRIGHT

TRADEMARKS

NAME

xauth – X authority file utility

SYNOPSIS

xauth [**-f** *authfile*] [**-v**] [**-q**] [**-i**] [**-b**] [*command arg...*]

DESCRIPTION

The **xauth** program is used to edit and display the authorization information used in connecting to the X server. This program is usually to extract authorization records from one machine and merge them in on another (as is the case when using remote logins or to grant access to other users). Commands (described below) may be entered interactively, on the **xauth** command line, or in scripts. Note that this program does **not** contact the X server.

OPTIONS

The following options may be used with **xauth.** They may be given individually (e.g. **–q –i**) or may combined (e.g. **-qi**):

–f *authfile*
: This option specifies the name of the authority file to use. By default, xauth **will use the file specified by the XAUTHORITY environment variable or** *.Xauthority* **in the user's home directory.**

–q
: This option indicates that **xauth** should operate quietly and not print unsolicited status messages. This is the default if an **xauth** command is is given on the command line or if the standard output is not directed to a terminal.

–v
: This option indicates that **xauth** should operate verbosely and print status messages indicating the results of various operations (e.g. how many records have been read in or written out). This is the default if **xauth** is reading commands from its standard input and its standard output is directed to a terminal.

–i
: This option indicates that **xauth** should ignore any authority file locks. Normally, **xauth** will refuse to read or edit any authority files that have been locked by other programs (usually **xdm**(1) or another **xauth**).

–b
: This option indicates that **xauth** should attempt to break any authority file locks before proceeding and should only be used to clean up stale locks.

COMMANDS

The following commands may be used to manipulate authority files:

add *displayname protocolname hexkey*
: An authorization entry for the indicated display using the given protocol and key data is added to the authorization file. The data is specified as an even-lengthed string of hexadecimal digits, each pair representing one octet. The first digit of each pair gives the most significant 4 bits of the octet and the second digit of the pairgives the least significant 4 bits. For example, a 32 character hexkey would represent a 128-bit value. A protocol name consisting of just a single period is treated as an abbreviation for *MIT-MAGIC-COOKIE-1*.

add *displayname protocolname 'string'*
: An authorization entry for the indicated display using the given protocol and the protocolname is specified by 'string'. If the authorization name is UN_DE-1, then the data field contains an ASCII netname.

[n]extract *filename displayname...*
: Authorization entries for each of the specified displays are written to the indicated file. If the *nextract* command is used, the entries are written in a numeric format suitable for non-binary transmission (such as secure electronic mail). The extracted entries can be read back in using the *merge* and *nmerge* commands. If the the filename consists of just a single dash, the entries will be written to the standard output.

[n]list [*displayname...]*
: Authorization entries for each of the specified displays (or all if no displays are named) are printed on the standard output. If the *nlist* command is used, entries will be shown in the numeric format used by the *nextract* command; otherwise, they are shown in a textual format. Key data is always displayed in the hexadecimal format given in the description of the *add* command.

[n]merge [*filename...]*
: Authorization entries are read from the specified files and are merged into the authorization database, superceding any matching existing entries. If the *nmerge* command is used, the numeric format given in the description of the *extract* command is used. If a filename consists of just a single dash, the standard input will be read if it hasn't been read before.

remove *displayname...*
: Authorization entries matching the specified displays are removed from the authority file.

source *filename*
: The specified file is treated as a script containing **xauth** commands to execute. Blank lines and lines beginning with a sharp sign (#) are ignored. A single dash may be used to indicate the standard input, if it hasn't already been read.

info Information describing the authorization file, whether or not any changes have been made, and from where **xauth** commands are being read is printed on the standard output.

exit If any modifications have been made, the authority file is written out (if allowed), and the program exits. An end of file is treated as an implicit *exit* command.

quit The program exits, ignoring any modifications. This may also be accomplished by pressing the interrupt character.

help [*string***]**
: A description of all commands that begin with the given string (or all commands if no string is given) is printed on the standard output.

? A short list of the valid commands is printed on the standard output.

DISPLAY NAMES

Display names for the *add*, *[n]extract*, *[n]list*, *[n]merge*, and *remove* commands use the same format as the DISPLAY environment variable and the common *-display* command line argument. Display-specific information (such as the screen number) is unnecessary and will be ignored. Same-machine connections (such as local-host sockets, shared memory, and the Internet Protocol hostname *localhost*) are referred to as *hostname*/unix:*displaynumber* so that local entries for different machines may be stored in one authority file.

EXAMPLE

The most common use for **xauth** is to extract the entry for the current display, copy it to another machine, and merge it into the user's authority file on the remote machine:

```
example%  xauth extract - $DISPLAY | rsh other xauth merge -
```

ENVIRONMENT

This **xauth** program uses the following environment variables:

XAUTHORITY
to get the name of the authority file to use if the *–f* option isn't used. If this variable is not set, **xauth** will use *.Xauthority* in the user's home directory.

HOME to get the user's home directory if XAUTHORITY isn't defined.

BUGS

Users that have unsecure networks should take care to use encrypted file transfer mechanisms to copy authorization entries between machines. Similarly, the *MIT-MAGIC-COOKIE-1* protocol is not very useful in unsecure environments. Sites that are interested in additional security may need to use encrypted authorization mechanisms such as Kerberos.

Spaces are currently not allowed in the protocol name. Quoting could be added for the truly perverse.

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xbiff – mailbox flag for X

SYNOPSIS

xbiff [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The **xbiff** program displays a little image of a mailbox. When there is no mail, the flag on the mailbox is down. When mail arrives, the flag goes up and the mailbox beeps. By default, pressing any mouse button in the image forces **xbiff** to remember the current size of the mail file as being the ‘‘empty’’ size and to lower the flag.

This program is nothing more than a wrapper around the Athena **Mailbox** widget.

OPTIONS

Xbiff accepts all of the standard X Toolkit command line options along with the additional options listed below:

–**help** This option indicates that a brief summary of the allowed options should be printed on the standard error.

–**update** *seconds*
This option specifies the frequency in seconds at which **xbiff** should update its display. If the mailbox is obscured and then exposed, it will be updated immediately. The default is 30 seconds.

–**file** *filename*
This option specifies the name of the file which should be monitored. By default, it watches **/var/mail/***username*, where *username* is your login name.

–**volume** *percentage*
This option specifies how loud the bell should be rung when new mail comes in.

–**shape** This option indicates that the mailbox window should be shaped if masks for the empty or full images are given.

The following standard X Toolkit command line arguments are commonly used with **xbiff:**

–**display** *display*
This option specifies the X server to contact.

–**geometry** *geometry*
This option specifies the preferred size and position of the mailbox window. The mailbox is 48 pixels wide and 48 pixels high and will be centered in the window.

–**bg** *color*
This option specifies the color to use for the background of the window.

–**bd** *color*
This option specifies the color to use for the border of the window.

–**bw** *number*
This option specifies the width in pixels of the border surrounding the window.

–**fg** *color*
This option specifies the color to use for the foreground of the window.

–rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–xrm *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

X DEFAULTS

The application class name is XBiff. This program uses the **Mailbox** widget in the Athena widget set. It understands all of the core resource names and classes as well as:

checkCommand (class **CheckCommand)**
Specifies a shell command to be executed to check for new mail rather than examining the size of **file**. The specified string value is used as the argument to a **system**(3) call and may therefore contain i/o redirection. An exit status of 0 indicates that new mail is waiting, 1 indicates that there has been no change in size, and 2 indicates that the mail has been cleared. By default, no shell command is provided.

file (class **File**)
Specifies the name of the file to monitor. The default is to watch **/var/mail/***username*, where *username* is your login name.

onceOnly
(class **Boolean**)
Specifies that the bell is only rung the first time new mail is found and is not rung again until at least one interval has passed with no mail waiting. The window will continue to indicate the presence of new mail until it has been retrieved. The default is false.

width (class **Width**)
Specifies the width of the mailbox.

height (class **Height**)
Specifies the height of the mailbox.

update (class **Interval**) Specifies the frequency in seconds at which the mail should be checked. The default is 30.

volume (class **Volume**)
Specifies how loud the bell should be rung. The default is 33 percent.

foreground
(class **Foreground**)
Specifies the color for the foreground.

reverseVideo
(class **ReverseVideo**)
Specifies that the foreground and background should be reversed.

flip (class **Flip**)
Specifies whether or not the image that is shown when mail has arrived should be inverted. The default is ‘‘true.’’

fullPixmap

(class **Pixmap**)
Specifies a bitmap to be shown when new mail has arrived. The default is flagup.

emptyPixmap
(class **Pixmap**)
Specifies a bitmap to be shown when no new mail is present. The default is flagdown.

shapeWindow
(class **ShapeWindow**)
Specifies whether or not the mailbox window should be shaped to the given fullPixmapMask and emptyPixmapMask. The default is false.

fullPixmapMask
(class **PixmapMask**)
Specifies a mask for the bitmap to be shown when new mail has arrived. The default is none.

emptyPixmapMask
(class **PixmapMask**)
Specifies a mask for the bitmap to be shown when no new mail is present. The default is none.

ACTIONS

The *Mailbox* widget provides the following actions for use in event translations:

check() This action causes the widget to check for new mail and display the flag appropriately.

unset() This action causes the widget to lower the flag until new mail comes in.

set() This action causes the widget to raise the flag until the user resets it.

The default translation is
```
     <ButtonPress>:  unset()
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X11(7), **xrdb**(1), **stat**(2)

BUGS

The mailbox bitmaps are ugly.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium
Additional hacks by Ralph Swick, DEC / MIT Project Athena

NAME

xcalc – scientific calculator for X

SYNOPSIS

xcalc [-stipple] [-rpn] [-*toolkitoption...*]

DESCRIPTION

xcalc is a scientific calculator desktop accessory that can emulate a TI-30 or an HP-10C.

OPTIONS

xcalc accepts all of the standard toolkit command line options along with two additional options:

–stipple
This option indicates that the background of the calculator should be drawn using a stipple of the foreground and background colors. On monochrome displays improves the appearance.

–rpn
This option indicates that Reverse Polish Notation should be used. In this mode the calculator will look and behave like an HP-10C. Without this flag, it will emulate a TI-30.

OPERATION

Pointer Usage: Operations may be performed with pointer button 1, or in some cases, with the keyboard. Many common calculator operations have keyboard accelerators. To quit, press pointer button 3 on the AC key of the TI calculator, or the ON key of the HP calculator.

Calculator Key Usage (TI mode): The numbered keys, the +/- key, and the +, -, *, /, and = keys all do exactly what you would expect them to. It should be noted that the operators obey the standard rules of precedence. Thus, entering "3+4*5=" results in "23", not "35". The parentheses can be used to override this. For example, "(1+2+3)*(4+5+6)=" results in "6*15=90".

The entire number in the calculator display can be selected, in order to paste the result of a calculation into text.

The action procedures associated with each function are given below. These are useful if you are interested in defining a custom calculator. The action used for all digit keys is **digit(***n*), where *n* is the corresponding digit, 0..9.

1/x
Replaces the number in the display with its reciprocal. The corresponding action procedure is **reciprocal()**.

x^2
Squares the number in the display. The corresponding action procedure is **square()**.

SQRT
Takes the square root of the number in the display. The corresponding action procedure is **squareRoot()**.

CE/C
When pressed once, clears the number in the display without clearing the state of the machine. Allows you to re-enter a number if you make a mistake. Pressing it twice clears the state, also. The corresponding action procedure for TI mode is **clear()**.

AC
Clears the display, the state, and the memory. Pressing it with the third pointer button turns off the calculator, in that it exits the program. The action

procedure to clear the state is **off()**; to quit, **quit()**.

INV	Invert function. See the individual function keys for details. The corresponding action procedure is **inverse()**.
sin	Computes the sine of the number in the display, as interpreted by the current DRG mode (see DRG, below). If inverted, it computes the arcsine. The corresponding action procedure is **sine()**.
cos	Computes the cosine, or arccosine when inverted. The corresponding action procedure is **cosine()**.
tan	Computes the tangent, or arctangent when inverted. The corresponding action procedure is **tangent()**.
DRG	Changes the DRG mode, as indicated by ’DEG’, ’RAD’, or ’GRAD’ at the bottom of of the calculator ‘‘liquid crystal’’ display. When in ’DEG’ mode, numbers in the display are taken as being degrees. In ’RAD’ mode, numbers are in radians, and in ’GRAD’ mode, numbers are in grads. When inverted, the DRG key has a feature of converting degrees to radians to grads and vice-versa. Example: put the calculator into ’DEG’ mode, and enter "45 INV DRG". The display should now show something along the lines of ".785398", which is 45 degrees converted to radians. The corresponding action procedure is **degree()**.
e	The constant ’e’. (2.7182818...). The corresponding action procedure is e().
EE	Used for entering exponential numbers. For example, to get "-2.3E-4" you’d enter "2 . 3 + / - EE 4 + / -". The corresponding action procedure is **scientific()**.
log	Calculates the log (base 10) of the number in the display. When inverted, it raises "10.0" to the number in the display. For example, entering "3 INV log" should result in "1000". The corresponding action procedure is **logarithm()**.
ln	Calculates the log (base e) of the number in the display. When inverted, it raises "e" to the number in the display. For example, entering "e ln" should result in "1". The corresponding action procedure is naturalLog().
y^x	Raises the number on the left to the power of the number on the right. For example "2 y^x 3 =" results in "8", which is 2^3. For a further example, "(1+2+3) y^x (1+2) =" equals "6 y^x 3" which equals "216". The corresponding action procedure is **power()**.
PI	The constant ’pi’. (3.1415927....) The corresponding action procedure is **pi()**.
x!	Computes the factorial of the number in the display. The number in the display must be an integer in the range 0-500, though, depending on your math library, it might overflow long before that. The corresponding action procedure is **factorial()**.
(	Left parenthesis. The corresponding action procedure for TI calculators is **leftParen()**.
)	Right parenthesis. The corresponding action procedure for TI calculators is **rightParen()**.

/	Division. The corresponding action procedure is **divide()**.
*	Multiplication. The corresponding action procedure is **multiply()**.
-	Subtraction. The corresponding action procedure is **subtract()**.
+	Addition. The corresponding action procedure is **add()**.
=	Perform calculation. The TI-specific action procedure is **equal()**.
STO	Copies the number in the display to the memory location. The corresponding action procedure is **store()**.
RCL	Copies the number from the memory location to the display. The corresponding action procedure is **recall()**.
SUM	Adds the number in the display to the number in the memory location. The corresponding action procedure is **sum()**.
EXC	Swaps the number in the display with the number in the memory location. The corresponding action procedure for the TI calculator is **exchange()**.
+/-	Negate; change sign. The corresponding action procedure is **negate()**.
.	Decimal point. The action procedure is **decimal()**.

Calculator Key Usage (RPN mode): The number keys, CHS (change sign), +, -, *, /, and ENTR keys all do exactly what you would expect them to do. Many of the remaining keys are the same as in TI mode. The differences are detailed below. The action procedure for the ENTR key is enter().

<-	This is a backspace key that can be used if you make a mistake while entering a number. It will erase digits from the display. (See BUGS). Inverse backspace will clear the X register. The corresponding action procedure is **back()**.
ON	Clears the display, the state, and the memory. Pressing it with the third pointer button turns off the calculator, in that it exits the program. To clear state, the action procedure is **off**; to quit, **quit()**.
INV	Inverts the meaning of the function keys. This would be the *f* key on an HP calculator, but *xcalc* does not display multiple legends on each key. See the individual function keys for details.
10^x	Raises "10.0" to the number in the top of the stack. When inverted, it calculates the log (base 10) of the number in the display. The corresponding action procedure is **tenpower()**.
e^x	Raises "e" to the number in the top of the stack. When inverted, it calculates the log (base e) of the number in the display. The action procedure is **epower()**.
STO	Copies the number in the top of the stack to a memory location. There are 10 memory locations. The desired memory is specified by following this key with a digit key.
RCL	Pushes the number from the specified memory location onto the stack.

SUM Adds the number on top of the stack to the number in the specified memory location.

x:y Exchanges the numbers in the top two stack positions, the X and Y registers. The corresponding action procedure is **XexchangeY()**.

R v Rolls the stack downward. When inverted, it rolls the stack upward. The corresponding action procedure is **roll()**.

blank These keys were used for programming functions on the HP-10C. Their functionality has not been duplicated in *xcalc*.

Finally, there are two additional action procedures: **bell()**, which rings the bell; and **selection()**, which performs a cut on the entire number in the calculator's ‘‘liquid crystal’’ display.

ACCELERATORS

Accelerators are shortcuts for entering commands. *xcalc* provides some sample keyboard accelerators; also users can customize accelerators. The numeric keypad accelerators provided by *xcalc* should be intuitively correct. The accelerators defined by *xcalc* on the main keyboard are given below:

TI Key	HP Key	Keyboard Accelerator	TI Function	HP Function
SQRT	SQRT	r	squareRoot()	squareRoot()
AC	ON	space	clear()	clear()
AC	<-	Delete	clear()	back()
AC	<-	Backspace	clear()	back()
AC	<-	Control-H	clear()	back()
AC		Clear	clear()	
AC	ON	q	quit()	quit()
AC	ON	Control-C	quit()	quit()
INV	i	i	inverse()	inverse()
sin	s	s	sine()	sine()
cos	c	c	cosine()	cosine()
tan	t	t	tangent()	tangent()
DRG	DRG	d	degree()	degree()
e		e	e()	
ln	ln	l	naturalLog()	naturalLog()
y^x	y^x	^	power()	power()
PI	PI	p	pi()	pi()
x!	x!	!	factorial()	factorial()
(		(	leftParen()	
)		)	rightParen()	
/	/	/	divide()	divide()
*	*	*	multiply()	multiply()
-	-	-	subtract()	subtract()

+	+	+	add()	add()
=		=	equal()	
0..9	0..9	0..9	digit()	digit()
.	.	.	decimal()	decimal()
+/-	CHS	n	negate()	negate()
	x:y	x		XexchangeY()
	ENTR	Return		enter()
	ENTR	Linefeed		enter()

CUSTOMIZATION

The application class name is XCalc.

xcalc has an enormous application defaults file which specifies the position, label, and function of each key on the calculator. It also gives translations to serve as keyboard accelerators. Because these resources are not specified in the source code, you can create a customized calculator by writing a private application defaults file, using the Athena Command and Form widget resources to specify the size and position of buttons, the label for each button, and the function of each button.

The foreground and background colors of each calculator key can be individually specified. For the TI calculator, a classical color resource specification might be:

```
XCalc.ti.Command.background: gray50
XCalc.ti.Command.foreground:  white
```

For each of buttons 20, 25, 30, 35, and 40, specify:

```
XCalc.ti.button20.background:   black
XCalc.ti.button20.foreground:   white
```

For each of buttons 22, 23, 24, 27, 28, 29, 32, 33, 34, 37, 38, and 39:

```
XCalc.ti.button22.background:   white
XCalc.ti.button22.foreground:   black
```

WIDGET HIERARCHY

In order to specify resources, it is useful to know the hierarchy of the widgets which compose xcalc. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XCalc xcalc
        Form  ti  or  hp    (the name depends on the mode)
                Form  bevel
                        Form  screen
                                Label  M
                                Toggle  LCD
                                Label  INV
                                Label  DEG
                                Label  RAD
```

```
                              Label  GRAD
                              Label  P
               Command  button1
               Command  button2
               Command  button3
```
and so on, ...
```
               Command  button38
               Command  button39
               Command  button40
```

APPLICATION RESOURCES

rpn (Class **Rpn**)
Specifies that the rpn mode should be used. The default is TI mode.

stipple (Class **Stipple**)
Indicates that the background should be stippled. The default is ''on'' for monochrome displays, and ''off'' for color displays.

cursor (Class **Cursor**)
The name of the symbol used to represent the pointer. The default is ''hand2''.

COLORS

If you would like xcalc to use its ti colors, include the following in the #ifdef COLOR section of the file you read with xrdb:

```
*customization:          -color
```

This will cause xcalc to pick up the colors in the app-defaults color customization file: /usr/openwin/lib/app-defaults/XCalc-color.

SEE ALSO

X11(7), **xrdb**(1), the Athena Widget Set

BUGS

HP mode: A bug report claims that the sequence of keys 5, ENTER, <- should clear the display, but it doesn't.

COPYRIGHT

Copyright 1988, 1989, Massachusetts Institute of Technology.
See **X11(7)** for a full statement of rights and permissions.

AUTHORS

John Bradley, University of Pennsylvania
Mark Rosenstein, MIT Project Athena
Donna Converse, MIT X Consortium

NAME

xclipboard – X clipboard client

SYNOPSIS

xclipboard [*-toolkitoption* ...] [-w] [-nw]

DESCRIPTION

The **xclipboard** program is used to collect and display text selections that are sent to the CLIPBOARD by other clients. It is typically used to save CLIPBOARD selections for later use. It stores each CLIPBOARD selection as a separate string, each of which can be selected. Each time CLIPBOARD is asserted by another application, **xclipboard** transfers the contents of that selection to a new buffer and displays it in the text window. Buffers are never automatically deleted, so you'll want to use the delete button to get rid of useless items.

Since **xclipboard** uses a Text Widget to display the contents of the clipboard, text sent to the CLIPBOARD may be re-selected for use in other applications. **xclipboard** also responds to requests for the CLIPBOARD selection from other clients by sending the entire contents of the currently displayed buffer.

An **xclipboard** window has the following buttons across the top:

quit When this button is pressed, **xclipboard** exits.

delete When this button is pressed, the current buffer is deleted and the next one displayed.

new Creates a new buffer with no contents. Useful in constructing a new CLIPBOARD selection by hand.

next Displays the next buffer in the list.

previous Displays the previous buffer.

OPTIONS

The **xclipboard** program accepts all of the standard X Toolkit command line options as well as the following:

–w This option indicates that lines of text that are too long to be displayed on one line in the clipboard should wrap around to the following lines.

–nw This option indicates that long lines of text should not wrap around. This is the default behavior.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **xclipboard**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XClipboard  xclipboard
        Form  form
                Command  quit
                Command  delete
                Command  new
                Command  next
                Command  prev
```

Text text

SENDING/RETRIEVING CLIPBOARD CONTENTS

Text is copied to the clipboard whenever a client asserts ownership of the **CLIPBOARD** selection. Text is copied from the clipboard whenever a client requests the contents of the **CLIPBOARD** selection. Examples of event bindings that a user may wish to include in a resource configuration file to use the clipboard are:

```
*VT100.Translations: #override \
        <Btn3Up>:              select-end(CLIPBOARD) \n\
        <Btn2Up>:              insert-selection(PRIMARY,CLIPBOARD) \n\
        <Btn2Down>:            ignore ()
```

SEE ALSO

X11(7), **xterm**(1), individual client documentation for how to make a selection and send it to the CLIPBOARD.

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/openwin/lib/app-defaults/XClipboard - specifies required resources

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology
See *X11(7)* for a full statement of rights and permissions.

AUTHOR

Ralph R. Swick, DEC / MIT Project Athena
Chris D. Peterson, MIT X Consortium
Keith Packard, MIT X Consortium

NAME

xclock – analog / digital clock for X

SYNOPSIS

xclock [*toolkitoption...*] [**–help**] [**–analog**] [**–digital**] [**–chime**] [**–hd** *color*] [**–hl** *color*] [**–update** *seconds*] [**–padding** *number*]

DESCRIPTION

The **xclock** program displays the time in analog or digital form. The time is continuously updated at a frequency which may be specified by the user. This program is nothing more than a wrapper around the Athena Clock widget.

OPTIONS

Xclock accepts all of the standard X Toolkit command line options along with the additional options listed below:

–help This option indicates that a brief summary of the allowed options should be printed on the standard error.

–analog This option indicates that a conventional 12 hour clock face with tick marks and hands should be used. This is the default.

–digital This option indicates that a 24 hour digital clock should be used.

–chime This option indicates that the clock should chime once on the half hour and twice on the hour.

–hd *color*
This option specifies the color of the hands on an analog clock. The default is *black.*

–hl *color*
This option specifies the color of the edges of the hands on an analog clock, and is only useful on color displays. The default is *black.*

–update *seconds*
This option specifies the frequency in seconds at which **xclock** should update its display. If the clock is obscured and then exposed, it will be updated immediately. A value of less than 30 seconds will enable a second hand on an analog clock. The default is 60 seconds.

–padding *number*
This option specifies the width in pixels of the padding between the window border and clock text or picture. The default is 10 on a digital clock and 8 on an analog clock.

X DEFAULTS

This program uses the *Athena Clock* widget. It understands all of the core resource names and classes as well as:

width (class **Width)**
Specifies the width of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

height (class **Height)**
Specifies the height of the clock. The default for analog clocks is 164 pixels; the

default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

update (class **Interval)**
Specifies the frequency in seconds at which the time should be redisplayed.

foreground (class **Foreground)**
Specifies the color for the tic marks. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *lwhite*, otherwise the default is *black*.

hands (class **Foreground)**
Specifies the color of the insides of the clock's hands. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *lwhite*, otherwise the default is *black*.

highlight (class **Foreground)**
Specifies the color used to highlight the clock's hands. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *lwhite*, otherwise the default is *black*.

analog (class **Boolean)**
Specifies whether or not an analog clock should be used instead of a digital one. The default is True.

chime (class **Boolean)**
Specifies whether or not a bell should be rung on the hour and half hour.

padding (class **Margin)**
Specifies the amount of internal padding in pixels to be used. The default is 8.

font (class **Font)**
Specifies the font to be used for the digital clock. Note that variable width fonts currently will not always display correctly.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **xclock**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XClock  xclock
        Clock  clock
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/openwin/lib/app-defaults/XClock **-** specifies required resources

SEE ALSO

X11(7), **xrdb**(1), **time**(1), Athena Clock widget

BUGS

Xclock believes the system clock.

When in digital mode, the string should be centered automatically.

COPYRIGHT

AUTHORS

Tony Della Fera (MIT-Athena, DEC)
Dave Mankins (MIT-Athena, BBN)
Ed Moy (UC Berkeley)

NAME

xcmsdb – Xlib Screen Color Characterization Data utility

SYNOPSIS

xcmsdb [**–query**] [**–remove**] [**–color**] [**–format 32 | 16 | 8**] [*filename*]

DESCRIPTION

xcmsdb is used to load, query, or remove Screen Color Characterization Data stored in properties on the root window of the screen. Screen Color Characterization Data is an integral part of Xlib, necessary for proper conversion between device-independent and device-dependent color specifications. Xlib uses the XDCCC_LINEAR_RGB_MATRICES and XDCCC_LINEAR_RGB_CORRECTION properties to store color characterization data for color monitors. It uses XDCCC_GRAY_SCREENWWHITEPOINT and XDCCC_GRAY_CORRECTION properties for gray scale monitors. Because Xlib allows the addition of Screen Color Characterization Function Sets, added function sets may place their Screen Color Characterization Data on other properties. This utility is unaware of these other properties, therefore, you will need to use a similar utility provided with the function set, or use the **xprop**(1) utility.

The ASCII readable contents of *filename* (or the standard input if no input file is given) are appropriately transformed for storage in properties, provided the **–query** or **–remove** options are not specified.

OPTIONS

xcmsdb program accepts the following options:

–query
This option attempts to read the XDCCC properties off the screen's root window. If successful, it transforms the data into a more readable format, then sends the data to standard out.

–remove
This option attempts to remove the XDCCC properties on the screen's root window.

–color
This option sets the query and remove options to only check for the XDCCC_LINEAR_RGB_MATRICES and XDCCC_LINEAR_RGB_CORRECTION properties. If the **–color** option is not set then the query and remove options check for all the properties.

–format 32 | 16 | 8
Specifies the property format (32, 16, or 8 bits per entry) for the XDCCC_LINEAR_RGB_CORRECTION property. Precision of encoded floating point values increases with the increase in bits per entry. The default is 32 bits per entry.

SEE ALSO

xprop(1), Xlib documentation

ENVIRONMENT

DISPLAY
to figure out which display and screen to use.

COPYRIGHT

Copyright 1990, Tektronix Inc.

AUTHOR

Chuck Adams, Tektronix Inc.

NAME

xcolor – displays 256 colors in an X window.

SYNOPSIS

xcolor [**–display** *display*] [**–geometry** *geometry*] [**–dump**] [**–nobw**] [**–half**]
[**–noinst**] [**–iconwin**]

DESCRIPTION

xcolor displays all 256 colors in a window. When you move the pointer into the window, installs a colormap containing a hue ramp with constant saturation and constant brightness, while preserving black and white.

OPTIONS

–display *connection*
Connect to X server display, *connection.*

–geometry *geomspec*
Set window size and placement to the standard X11 geometry specification, *geomspec.*

–dump
Dumps the RGB values of the default colormap to stdout.

–nobw
Don't preserve black and white in the hue ramp.

–half
Create the hue ramp using only the upper half of the colormap.

–noinst
Don't install the hue ramp.

–iconwin
Use as miniature version of the main window as the icon.

COPYRIGHT

NAME

xconsole – monitor system console messages

SYNOPSIS

xconsole [*toolkitoption...*] [**–file** *filename*] [**–notify**] [**–stripNonprint**] [**–daemon**] [**–verbose**] [**–exitOnFail**]

DESCRIPTION

The **xconsole** program displays messages which are usually sent to /dev/console.

OPTIONS

Xconsole accepts all of the standard X Toolkit command line options along with the additional options listed below:

–file *file-name*
To monitor some other device, use this option to specify the device name. This does not work on regular files as they are always ready to be read from.

–notify –nonotify
When new data are received from the console and the notify option is set, the icon name of the application has " *" appended, so that it is evident even when the application is iconified. –notify is the default.

–daemon
This option causes **xconsole** to place itself in the background, using fork/exit.

–verbose
When set, this option directs **xconsole** to display an informative message in the first line of the text buffer.

–exitOnFail
When set, this option directs **xconsole** to exit when it is unable to redirect the console output.

X DEFAULTS

This program uses the *Athena Text* widget, look in the *Athena Widget Set* documentation for controlling it.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **xconsole**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XConsole  xconsole
        XConsole  text
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/openwin/lib/app-defaults/XConsole **-** specifies required resources

SEE ALSO

X11(7), **xrdb**(1), Athena Text widget

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Keith Packard (MIT X Consortium)

NAME

xcutsel – interchange between cut buffer and selection

SYNOPSIS

xcutsel [-*toolkitoption...*] [**-selection** *selection*] [**-cutbuffer** *number*]

DESCRIPTION

The **xcutsel** program is used to copy the current selection into a cut buffer and to make a selection that contains the current contents of the cut buffer. It acts as a bridge between applications that don't support selections and those that do.

By default, **xcutsel** will use the selection named PRIMARY and the cut buffer CUT_BUFFER0. Either or both of these can be overridden by command line arguments or by resources.

An **xcutsel** window has the following buttons:

quit When this button is pressed, **xcutsel** exits. Any selections held by **xcutsel** are automatically released.

copy PRIMARY to 0
When this button is pressed, **xcutsel** copies the current selection into the cut buffer.

copy 0 to PRIMARY
When this button is pressed, **xcutsel** converts the current contents of the cut buffer into the selection.

The button labels reflect the selection and cutbuffer selected by command line options or through the resource database.

When the ‘‘copy 0 to PRIMARY’’ button is activated, the button will remain inverted as long as **xcutsel** remains the owner of the selection. This serves to remind you which client owns the current selection. Note that the value of the selection remains constant; if the cutbuffer is changed, you must again activate the copy button to retrieve the new value when desired.

OPTIONS

xcutsel accepts all of the standard X Toolkit command line options as well as the following:

–**selection** *name*
This option specifies the name of the selection to use. The default is PRIMARY. The only supported abbreviations for this option are ‘‘-select’’, ‘‘-sel’’ and ‘‘-s’’, as the standard toolkit option ‘‘-selectionTimeout’’ has a similar name.

–**cutbuffer** *number*
This option specifies the cut buffer to use. The default is cut buffer 0.

X DEFAULTS

This program accepts all of the standard X Toolkit resource names and classes as well as:

selection (class **Selection)**
This resource specifies the name of the selection to use. The default is PRIMARY.

cutBuffer (class **CutBuffer)**
This resource specifies the number of the cut buffer to use. The default is 0.

WIDGET NAMES

The following instance names may be used when user configuration of the labels in them is desired:

sel-cut (class **Command)**
This is the ‘‘copy SELECTION to BUFFER’’ button.

cut-sel (class **Command)**
This is the ‘‘copy BUFFER to SELECTION’’ button.

quit (class **Command)**
This is the ‘‘quit’’ button.

SEE ALSO

xclipboard(1), **xterm**(1)
text widget documentation, individual client documentation for how to make a selection.

BUGS

There is no way to change the name of the selection or the number of the cut buffer while the program is running.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Ralph R. Swick, DEC / MIT Project Athena

NAME

xditview – display ditroff DVI files

SYNOPSIS

xditview [*–toolkitoption* . . .] [*–option* . . .] [*filename*]

DESCRIPTION

The **xditview** program displays *ditroff* output on an X display. It uses no special metrics and automatically converts the printer coordinates into screen coordinates; using the user-specified screen resolution, rather than the actual resolution so that the appropriate fonts can be found. If ''–'' is given as the *filename*, **xditview** reads from standard input. If ''**|**'' is the first character of *filename*, **xditview** forks *sh* to run the rest of the ''file name'' and uses the standard output of that command.

OPTIONS

Xditview accepts all of the standard X Toolkit command line options along with the additional options listed below:

–**page** *page-number*
: This option specifies the page number of the document to be displayed at start up time.

–**resolution** *screen-resolution*
: This specifies the desired screen resolution to use; fonts will be opened by requesting this resolution field in the XLFD names.

–**noPolyText**
: Some X servers incorrectly implement PolyText with multiple strings per request. This option suppesses the use of this feature in **xditview.**

–**backingStore** *backing-store-type*
: Redisplay of the DVI window can take up to a second or so, this option causes the server to save the window contents so that when it is scrolled around the viewport, the window is painted from contents saved in backing store. *backing-store-type* can be one of **Always**, WhenMapped or NotUseful.

The following standard X Toolkit command line arguments are commonly used with **xditview:**

–**bg** *color*
: This option specifies the color to use for the background of the window. The default is *white.*

–**bd** *color*
: This option specifies the color to use for the border of the window. The default is *black.*

–**bw** *number*
: This option specifies the width in pixels of the border surrounding the window.

–**fg** *color*
: This option specifies the color to use for displaying text. The default is *black.*

–**fn** *font* This option specifies the font to be used for displaying widget text. The default is *fixed.*

–**rv** This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–geometry *geometry*
This option specifies the preferred size and position of the window.

–display *host*:*display*
This option specifies the X server to contact.

–xrm *resourcestring*
This option specifies a resource string to be used.

X DEFAULTS

This program uses a *Dvi* widget. It understands all of the core resource names and classes as well as:

width (class **Width)**
Specifies the width of the window.

height (class **Height)**
Specifies the height of the window.

foreground (class **Foreground)**
Specifies the default foreground color.

font (class **Font)**
Specifies the font to be used for error messages.

FontMap (class FontMap)
To associate the *ditroff* fonts with appropriate X fonts, this string resource contains a set of new-line separated specifications, each of which consists of a ditroff name, some white space and an XLFD pattern with ∗ characters in appropriate places to allow all sizes to be listed. The default fontMap is:

```
R     –∗–times–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
I     –∗–times–medium–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
B     –∗–times–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
F     –∗–times–bold–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
BI    –∗–times–bold–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
C     –∗–courier–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
CO    –∗–courier–medium–o–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
CB    –∗–courier–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
CF    –∗–courier–bold–o–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
H     –∗–helvetica–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
HO    –∗–helvetica–medium–o–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
HB    –∗–helvetica–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
HF    –∗–helvetica–bold–o–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
N     –∗–new century schoolbook–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
NI    –∗–new century schoolbook–medium–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
NB    –∗–new century schoolbook–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
NF    –∗–new century schoolbook–bold–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
A     –∗–charter–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
AI    –∗–charter–medium–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1 \ n \
AB      –*–charter–bold–r–normal––*–*–*–*–*–*–iso8859–1 \ n \
AF      –*–charter–bold–i–normal––*–*–*–*–*–*–iso8859–1 \ n \
S       –*–symbol–medium–r–normal––*–*–*–*–*–*–adobe–fontspecific \ n \
S2      –*–symbol–medium–r–normal––*–*–*–*–*–*–adobe–fontspecific \ n
```

USING XDITVIEW WITH DITROFF

You can use any DVI file with **xditview,** although DVI files which use the fonts appropriate to the fontMap will look more accurate on the screen. On servers which support scaled fonts, all requested font sizes will be accurately reflected on the screen; for servers which do not support scaled **xditview** will use the closest font from the same family.

SEE ALSO

X11(7), **xrdb**(1), *X Logical Font Description Conventions*

ORIGIN

Portions of this program originated in *xtroff which was derived* from *suntroff.*

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHORS

Keith Packard (MIT X Consortium)
Richard L. Hyde (Purdue)
David Slattengren (Berkeley)
Malcom Slaney (Schlumberger Palo Alto Research)
Mark Moraes (University of Toronto)

NAME

xdm – X Display Manager with support for XDMCP

SYNOPSIS

xdm [**–config** *configuration_file*] [**–nodaemon**] [**–debug** *debug_level*] [**–error** *error_log_file*] [**–resources** *resource_file*] [**–server** *server_entry*] [**–session** *session_program*]

DESCRIPTION

Xdm manages a collection of X displays, which may be on the local host or remote servers. The design of **xdm** was guided by the needs of X terminals as well as the X Consortium standard XDMCP, the *X Display Manager Control Protocol*. **Xdm** provides services similar to those provided by **init**(1M)**, getty**(1M), and **login**(1) on character terminals: prompting for login name and password, authenticating the user, and running a ‘‘session.’’

A ‘‘session’’ is defined by the lifetime of a particular process; in the traditional character-based terminal world, it is the user’s login shell. In the **xdm** context, it is an arbitrary session manager. This is because in a windowing environment, a user’s login shell process does not necessarily have any terminal-like interface with which to connect. When a real session manager is not available, a window manager or terminal emulator is typically used as the ‘‘session manager,’’ meaning that termination of this process terminates the user’s session.

When the session is terminated, **xdm** resets the X server and (optionally) restarts the whole process.

When **xdm** receives an Indirect query via XDMCP, it can run a chooser **process to perform an XDMCP BroadcastQuery (or an XDMCP Query to specified hosts) on behalf of the display and offer a menu of possible hosts that offer XDMCP display management. This feature is useful with X terminals that do not offer a host menu themselves.**

Because **xdm** provides the first interface that users will see, it is designed to be simple to use and easy to customize to the needs of a particular site. **Xdm** has many options, most of which have reasonable defaults. Browse through the various sections of this manual, picking and choosing the things you want to change. Pay particular attention to the **Session Program** section, which will describe how to set up the style of session desired.

TYPICAL USAGE

Actually, **xdm** is designed to operate in such a wide variety of environments that *typical* is probably a misnomer.

First, the **xdm** configuration file should be set up. Make a directory (usually */usr/openwin/lib/xdm*) to contain all of the relevant files. Here is a reasonable configuration file, which could be named *xdm-config*:

```
DisplayManager.servers:          /usr/openwin/lib/xdm/Xservers
DisplayManager.errorLogFile:     /usr/tmp/xdm-errors
DisplayManager*resources:        /usr/openwin/lib/xdm/Xresources
DisplayManager*startup:          /usr/openwin/lib/xdm/Xstartup
```

```
DisplayManager*session:           /usr/openwin/lib/xdm/Xsession
DisplayManager.pidFile:           /usr/tmp/xdm-pid
DisplayManager._0.authorize:      true
DisplayManager*authorize:         false
```

Note that this file simply contains references to other files. Note also that some of the resources are specified with ''*'' separating the components. These resources can be made unique for each different display, by replacing the ''*'' with the display-name, but normally this is not very useful. See the **Resources** section for a complete discussion.

The first file, */usr/openwin/lib/xdm/Xservers*, contains the list of displays to manage that are not using XDMCP. Most workstations have only one display, numbered 0, so the file will look something like this:

```
:0 Local local /usr/openwin/bin/X :0
```

This will keep */usr/openwin/bin/X* running on this display and manage a continuous cycle of sessions.

The file */usr/tmp/xdm-errors* will contain error messages from **xdm** and anything output to stderr by *Xsetup, Xstartup, Xsession* or *Xreset*. When you have trouble getting **xdm** working, check this file to see if **xdm** has any clues to the trouble.

The next configuration entry, */usr/openwin/lib/xdm/Xresources*, is loaded onto the display as a resource database using **xrdb**(1). As the authentication widget reads this database before starting up, it usually contains parameters for that widget:

```
xlogin*login.translations: #override\
        <Key>F1: set-session-argument(failsafe) finish-field()\n\
        <Key>Return: set-session-argument() finish-field()
xlogin*borderWidth: 3
#ifdef COLOR
xlogin*greetColor: CadetBlue
xlogin*failColor: red
#endif
```

Please note the translations entry; it specifies a few new translations for the widget which allow users to escape from the default session (and avoid troubles that may occur in it). Note that if #override is not specified, the default translations are removed and replaced by the new value, not a very useful result as some of the default translations are quite useful (such as ''<Key>: insert-char ()'' which responds to normal typing).

The *Xstartup* file shown here simply prevents login while the file */etc/nologin* exists. As there is no provision for displaying any messages here (there isn't any core X client which displays files), the user will probably be baffled by this behavior. Thus this is not a complete example, but simply a demonstration of the available functionality.

Here is a sample *Xstartup* script:

```
#!/bin/sh
#
# Xstartup
#
# This program is run as root after the user is verified
#
if [ –f /etc/nologin ]; then
        exit 1
fi
exit 0
```

The most interesting script is *Xsession*. This version recognizes the special ‘‘failsafe’’ mode, specified in the translations in the *Xresources* file above, to provide an escape from the ordinary session:

```
#!/bin/sh
#
# Xsession
#
# This is the program that is run as the client
# for the display manager. This example is
# quite friendly as it attempts to run a per-user
# .xsession file instead of forcing a particular
# session layout
#

case $# in
1)
        case $1 in
        failsafe)
                exec xterm –geometry 80x24–0–0 –ls
                ;;
        esac
esac

startup=$HOME/.xsession
resources=$HOME/.Xresources

if [ –f $startup ]; then
        exec $startup
        exec /bin/sh $startup
else
        if [ ! –f $resources ]; then
                resources=$HOME/.Xdefaults
        fi
        if [ –f $resources ]; then
                xrdb –load $resources
        fi
        twm &
        exec xterm –geometry 80x24+10+10 –ls
fi
```

OPTIONS

All of these options, except **–config**, specify values which can also be specified in the configuration file as resources.

–config *configuration_file*
: Names the configuration file, which specifies resources to control the behavior of **xdm.** */usr/openwin/lib/xdm/xdm-config* is the default.

–nodaemon
: Specifies ''false'' as the value for the **DisplayManager.daemonMode** resource. This suppresses the normal daemon behavior, which is for **xdm** to close all file descriptors, disassociate itself from the controlling terminal, and put itself in the background when it first starts up.

–debug *debug_level*
: Specifies the numeric value for the **DisplayManager.debugLevel** resource. A non-zero value causes **xdm** to print lots of debugging statements to the terminal; it also disables the **DisplayManager.daemonMode** resource, forcing **xdm** to run synchronously. To interpret these debugging messages, a copy of the source code for **xdm** is almost a necessity. No attempt has been made to rationalize or standardize the output.

–error *error_log_file*
: Specifies the value for the **DisplayManager.errorLogFile** resource. This file contains errors from **xdm** as well as anything written to stderr by the various scripts and programs run during the progress of the session.

–resources *resource_file*
: Specifies the value for the **DisplayManager*resources** resource. This file is loaded using **xrdb**(1) to specify configuration parameters for the authentication widget.

–server *server_entry*
: Specifies the value for the **DisplayManager.servers** resource. See the section **Server Specification** for a description of this resource.

–udpPort *port_number*
: Specifies the value for the **DisplayManager.requestPort** resource. This sets the port-number which **xdm** will monitor for XDMCP requests. As XDMCP uses the registered well-known UDP port 177, this resource should not be changed except for debugging.

–session session_program
: Specifies the value for the **DisplayManager*session** resource. This indicates the

program to run as the session after the user has logged in.

–xrm resource_specification
Allows an arbitrary resource to be specified, as in most X Toolkit applications.

RESOURCES

At many stages the actions of **xdm** can be controlled through the use of its configuration file, which is in the X resource format. Some resources modify the behavior of **xdm** on all displays, while others modify its behavior on a single display. Where actions relate to a specific display, the display name is inserted into the resource name between ''Display-Manager'' and the final resource name segment. For example, **DisplayManager.expo_0.startup** is the name of the resource which defines the startup shell file on the ''expo:0'' display. Because the resource manager uses colons to separate the name of the resource from its value and dots to separate resource name parts, **xdm** substitutes underscores for both dots and colons when generating the resource name.

DisplayManager.servers
This resource either specifies a file name full of server entries, one per line (if the value starts with a slash), or a single server entry. See the section **Server Specification** for the details.

DisplayManager.requestPort
This indicates the UDP port number which **xdm** uses to listen for incoming XDMCP requests. Unless you need to debug the system, leave this with its default value of 177.

DisplayManager.errorLogFile
Error output is normally directed at the system console. To redirect it, set this resource to a file name. A method to send these messages to *syslog* should be developed for systems which support it; however, the wide variety of interfaces precludes any system-independent implementation. This file also contains any output directed to stderr by the *Xsetup, Xstartup, Xsession* and *Xreset* files, so it will contain descriptions of problems in those scripts as well.

DisplayManager.debugLevel
If the integer value of this resource is greater than zero, reams of debugging information will be printed. It also disables daemon mode, which would redirect the information into the bit-bucket, and allows non-root users to run **xdm,** which would normally not be useful.

DisplayManager.daemonMode
Normally, **xdm** attempts to make itself into a daemon process unassociated with any terminal. This is accomplished by forking and leaving the parent process to exit, then closing file descriptors and releasing the controlling terminal. In some environments this is not desired (in particular, when debugging). Setting this resource to ''false'' will disable this feature.

DisplayManager.pidFile
The filename specified will be created to contain an ASCII representation of the process-id of the main **xdm** process. **Xdm** also uses file locking on this file to attempt to eliminate multiple daemons running on the same machine, which

would cause quite a bit of havoc.

DisplayManager.lockPidFile
This is the resource which controls whether **xdm** uses file locking to keep multiple display managers from running amok. On System V, this uses the **lockf**(3C) library call, while on BSD it uses **flock**(3B)

DisplayManager.remoteAuthDir
This names a directory in which **xdm** stores authorization files while initializing the session. The default value is */usr/openwin/lib/xdm.*

DisplayManager.autoRescan
This boolean controls whether **xdm** rescans the configuration, servers, access control and authentication keys files after a session terminates and the files have changed. By default it is ''true.'' You can force **xdm** to reread these files by sending a SIGHUP to the main process.

DisplayManager.removeDomainname
When computing the display name for XDMCP clients, the name resolver will typically create a fully qualified host name for the terminal. As this is sometimes confusing, **xdm** will remove the domain name portion of the host name if it is the same as the domain name of the local host when this variable is set. By default the value is ''true.''

DisplayManager.keyFile
XDM-AUTHENTICATION-1 style XDMCP authentication requires that a private key be shared between **xdm** and the terminal. This resource specifies the file containing those values. Each entry in the file consists of a display name and the shared key. By default, **xdm** does not include support for XDM-AUTHENTICATION-1, as it requires DES which is not generally distributable because of United States export restrictions.

DisplayManager.accessFile
To prevent unauthorized XDMCP service and to allow forwarding of XDMCP IndirectQuery requests, this file contains a database of hostnames which are either allowed direct access to this machine, or have a list of hosts to which queries should be forwarded to. The format of this file is described in the section **XDMCP Access Control.**

DisplayManager.DISPLAY.resources
This resource specifies the name of the file to be loaded by **xrdb**(3) as the resource database onto the root window of screen 0 of the display. The *Xsetup* program, the Login widget, and *chooser* will use the resources set in this file. This resource data base is loaded just before the authentication procedure is started, so it can control the appearance of the login window. See the section **Authentication Widget,** which describes the various resources that are appropriate to place in this file. There is no default value for this resource, but */usr/openwin/lib/xdm/Xresources* is the conventional name.

DisplayManager.DISPLAY.chooser
Specifies the program run to offer a host menu for Indirect queries redirected to

the special host name CHOOSER. */usr/openwin/lib/xdm/chooser* is the default. See the sections **XDMCP Access Control** and **Chooser**.

DisplayManager.DISPLAY.xrdb
Specifies the program used to load the resources. By default, **xdm** uses */usr/openwin/bin/xrdb.*

DisplayManager.DISPLAY.cpp
This specifies the name of the C preprocessor which is used by **xrdb**(1).

DisplayManager.DISPLAY.setup
This specifies a program which is run (as root) before offering the Login window. This may be used to change the appearence of the screen around the Login window or to put up other windows (e.g., you may want to run *xconsole* here). By default, no program is run. The conventional name for a file used here is *Xsetup*. See the section **Setup Program.**

DisplayManager.DISPLAY.startup
This specifies a program which is run (as root) after the authentication process succeeds. By default, no program is run. The conventional name for a file used here is *Xstartup*. See the section **Startup Program.**

DisplayManager.DISPLAY.session
This specifies the session to be executed (not running as root). By default, */usr/openwin/bin/xterm* is run. The conventional name is *Xsession*. See the section **Session Program.**

DisplayManager.DISPLAY.reset
This specifies a program which is run (as root) after the session terminates. Again, by default no program is run. The conventional name is *Xreset*. See the section **Reset Program.**

DisplayManager.DISPLAY.openDelay

DisplayManager.DISPLAY.openRepeat

DisplayManager.DISPLAY.openTimeout

DisplayManager.DISPLAY.startAttempts
These numeric resources control the behavior of **xdm** when attempting to open intransigent servers. **openDelay** is the length of the pause (in seconds) between successive attempts, **openRepeat** is the number of attempts to make, **openTimeout** is the amount of time to wait while actually attempting the open (i.e., the maximum time spent in the *connect*(2) system call) and **startAttempts** is the number of times this entire process is done before giving up on the server. After **openRepeat** attempts have been made, or if **openTimeout** seconds elapse in any particular attempt, **xdm** terminates and restarts the server, attempting to connect again. This process is repeated **startAttempts** times, at which point the display is declared dead and disabled. Although this behavior may seem arbitrary, it has been empirically developed and works quite well on most systems. The default values are 5 for **openDelay**, 5 for **openRepeat**, 30 for **openTimeout** and 4 for **startAttempts**.

DisplayManager.DISPLAY.pingInterval

DisplayManager.DISPLAY.pingTimeout
To discover when remote displays disappear, **xdm** occasionally pings them, using an X connection and *XSync* calls. **pingInterval** specifies the time (in minutes) between each ping attempt, **pingTimeout** specifies the maximum amount of time (in minutes) to wait for the terminal to respond to the request. If the terminal does not respond, the session is declared dead and terminated. By default, both are set to 5 minutes. If you frequently use X terminals which can become isolated from the managing host, you may wish to increase this value. The only worry is that sessions will continue to exist after the terminal has been accidentally disabled. **xdm** will not ping local displays. Although it would seem harmless, it is unpleasant when the workstation session is terminated as a result of the server hanging for NFS service and not responding to the ping.

DisplayManager.DISPLAY.terminateServer
This boolean resource specifies whether the X server should be terminated when a session terminates (instead of resetting it). This option can be used when the server tends to grow without bound over time, in order to limit the amount of time the server is run. The default value is ‘‘false.’’

DisplayManager.DISPLAY.userPath
Xdm sets the PATH environment variable for the session to this value. It should be a colon separated list of directories; see **sh**(1) for a full description. ‘‘:/bin:/usr/bin:/usr/openwin/bin:/usr/ucb’’ is a common setting. The default value can be specified at build time in the X system configuration file with DefaultUserPath;

DisplayManager.DISPLAY.systemPath
Xdm sets the PATH environment variable for the startup and reset scripts to the value of this resource. The default for this resource is specified at build time by the DefaultSystemPath entry in the system configuration file; ‘‘/etc:/bin:/usr/bin:/usr/openwin/bin:/usr/ucb’’ is a common choice. Note the absence of ‘‘.’’ from this entry. This is a good practice to follow for root; it avoids many common Trojan Horse system penetration schemes.

DisplayManager.DISPLAY.systemShell
Xdm sets the SHELL environment variable for the startup and reset scripts to the value of this resource. It is */bin/sh* by default.

DisplayManager.DISPLAY.failsafeClient
If the default session fails to execute, **xdm** will fall back to this program. This program is executed with no arguments, but executes using the same environment variables as the session would have had (see the section **Session Program**). By default, */usr/openwin/bin/xterm* is used.

DisplayManager.DISPLAY.grabServer

DisplayManager.DISPLAY.grabTimeout
To improve security, **xdm** grabs the server and keyboard while reading the login name and password. The **grabServer** resource specifies if the server should be

held for the duration of the name / password reading. When ''false,'' the server is ungrabbed after the keyboard grab succeeds, otherwise the server is grabbed until just before the session begins. The default is ''false.'' The **grabTimeout** resource specifies the maximum time **xdm** will wait for the grab to succeed. The grab may fail if some other client has the server grabbed, or possibly if the network latencies are very high. This resource has a default value of 3 seconds; you should be cautious when raising it, as a user can be spoofed by a look-alike window on the display. If the grab fails, **xdm** kills and restarts the server (if possible) and the session.

DisplayManager.DISPLAY.authorize

DisplayManager.DISPLAY.authName

authorize is a boolean resource which controls whether **xdm** generates and uses authorization for the local server connections. If authorization is used, **authName** specifies the type to use. Currently, **xdm** supports only MIT-MAGIC-COOKIE-1 authorization. XDM-AUTHORIZATION-1 could be supported as well, but DES is not generally distributable. XDMCP connections specify which authorization types are supported dynamically, so **authName** is ignored in this case. When **authorize** is set for a display and authorization is not available, the user is informed by having a different message displayed in the login widget. By default, **authorize** is ''true''; **authName** is ''MIT-MAGIC-COOKIE-1.''

DisplayManager.DISPLAY.authFile

This file is used to communicate the authorization data from **xdm** to the server, using the **–auth** server command line option. It should be kept in a directory which is not world-writable as it could easily be removed, disabling the authorization mechanism in the server.

DisplayManager.DISPLAY.resetForAuth

The original implementation of authorization in the sample server reread the authorization file at server reset time, instead of when checking the initial connection. As **xdm** generates the authorization information just before connecting to the display, an old server would not get up-to-date authorization information. This resource causes **xdm** to send SIGHUP to the server after setting up the file, causing an additional server reset to occur, during which time the new authorization information will be read. The default is ''false,'' which will work for all MIT servers.

DisplayManager.DISPLAY.userAuthDir

When **xdm** is unable to write to the usual user authorization file ($HOME / .Xauthority), it creates a unique file name in this directory and points the environment variable XAUTHORITY at the created file. It uses */tmp* by default.

XDMCP ACCESS CONTROL

The database file specified by the **DisplayManager.accessFile** provides information which **xdm** uses to control access from displays requesting XDMCP service. This file contains three types of entries: entries which control the response to Direct and Broadcast queries, entries which control the response to Indirect queries, and macro definitions.

The format of the Direct entries is simple, either a host name or a pattern, which is distinguished from a host name by the inclusion of one or more meta characters ('∗' matches any sequence of 0 or more characters, and '?' matches any single character) which are compared against the host name of the display device. If the entry is a host name, all comparisons are done using network addresses, so any name which converts to the correct network address may be used. For patterns, only canonical host names are used in the comparison, so ensure that you do not attempt to match aliases. Preceding either a host name or a pattern with a '!' character causes hosts which match that entry to be excluded.

An Indirect entry also contains a host name or pattern, but follows it with a list of host names or macros to which indirect queries should be sent.

A macro definition contains a macro name and a list of host names and other macros that the macro expands to. To distinguish macros from hostnames, macro names start with a '%' character. Macros may be nested.

Indirect entries may also specify to have **xdm** run *chooser* to offer a menu of hosts to connect to. See the section **Chooser**.

When checking access for a particular display host, each entry is scanned in turn and the first matching entry determines the response. Direct and Broadcast entries are ignored when scanning for an Indirect entry and vice-versa.

Blank lines are ignored, '#' is treated as a comment delimiter causing the rest of that line to be ignored, and '*newline*' causes the newline to be ignored, allowing indirect host lists to span multiple lines.

Here is an example Xaccess file:

```
#
# Xaccess – XDMCP access control file
#

#
# Direct/Broadcast query entries
#

!xtra.lcs.mit.edu          # disallow direct/broadcast service for xtra
bambi.ogi.edu              # allow access from this particular display
∗.lcs.mit.edu              # allow access from any display in LCS

#
# Indirect query entries
#

%HOSTS                     expo.lcs.mit.edu xenon.lcs.mit.edu \
                           excess.lcs.mit.edu kanga.lcs.mit.edu

extract.lcs.mit.edu        xenon.lcs.mit.edu          #force extract to contact xenon
```

```
!xtra.lcs.mit.edu         dummy               #disallow indirect access
*.lcs.mit.edu             %HOSTS              #all others get to choose
```

CHOOSER

For X terminals that do not offer a host menu for use with Broadcast or Indirect queries, the *chooser* program can do this for them. In the *Xaccess* file, specify ‘‘CHOOSER’’ as the first entry in the Indirect host list. *Chooser* will send a Query request to each of the remaining host names in the list and offer a menu of all the hosts that respond.

The list may consist of the word ‘‘BROADCAST,’’ in which case *chooser* will send a Broadcast instead, again offering a menu of all hosts that respond. Note that on some operating systems, UDP packets cannot be broadcast, so this feature will not work.

Example *Xaccess* file using *chooser*:

```
extract.lcs.mit.edu       CHOOSER %HOSTS      #offer a menu of these hosts
xtra.lcs.mit.edu          CHOOSER BROADCAST   #offer a menu of all hosts
```

The program to use for **chooser** is specified by the **DisplayManager.DISPLAY.chooser** resource. Resources for this program can be put into the file named by **DisplayManager.DISPLAY.resources**.

SERVER SPECIFICATION

The resource **DisplayManager.servers** gives a server specification or, if the values starts with a slash (/), the name of a file containing server specifications, one per line.

Each specification indicates a display which should constantly be managed and which is not using XDMCP. Each consists of at least three parts: a display name, a display class, a display type, and (for local servers) a command line to start the server. A typical entry for local display number 0 would be:

```
  :0 local /usr/openwin/bin/X :0
```

The display types are:

local	local display: **xdm** must run the server
foreign	remote display: **xdm** opens an X connection to a running server

The display name must be something that can be passed in the **–display** option to an X program. This string is used to generate the display-specific resource names, so be careful to match the names (e.g. use ‘‘:0 local /usr/openwin/bin/X :0’’ instead of ‘‘localhost:0 local /usr/openwin/bin/X :0’’ if your other resources are specified as ‘‘DisplayManager._0.session’’). The display class portion is also used in the display-specific resources, as the class of the resource. This is useful if you have a large collection of similar displays (like a corral of X terminals) and would like to set resources for groups of them. When using XDMCP, the display is required to specify the display class, so the manual for your particular X terminal should document the display class string for your device. If it doesn’t, you can run **xdm** in debug mode and look at the resource strings which it generates for that device, which will include the class string.

SETUP PROGRAM

The *Xsetup* file is run after the server is reset, but before the Login window is offered. The file is typically a shell script. It is run as root, so should be careful about security. This is the place to change the root background or bring up other windows that should appear on the screen along with the Login widget. Note that since **xdm** grabs the keyboard, any other windows will not be able to receive keyboard input. They will be able to interact with the mouse, however; beware of potential security holes here. If **DisplayManager.DISPLAY.grabServer** is set, *Xsetup* will not be able to connect to the display at all. Resources for this program can be put into the file named by **DisplayManager.DISPLAY.resources**.

AUTHENTICATION WIDGET

The authentication widget reads a name/password pair from the keyboard. Nearly every imaginable parameter can be controlled with a resource. Resources for this widget should be put into the file named by **DisplayManager.DISPLAY.resources**. All of these have reasonable default values, so it is not necessary to specify any of them.

xlogin.Login.width, xlogin.Login.height, xlogin.Login.x, xlogin.Login.y
The geometry of the Login widget is normally computed automatically. If you wish to position it elsewhere, specify each of these resources.

xlogin.Login.foreground
The color used to display the typed-in user name.

xlogin.Login.font
The font used to display the typed-in user name.

xlogin.Login.greeting
A string which identifies this window. The default is ‘‘X Window System.’’

xlogin.Login.unsecureGreeting
When X authorization is requested in the configuration file for this display and none is in use, this greeting replaces the standard greeting. The default is ‘‘This is an unsecure session’’

xlogin.Login.greetFont
The font used to display the greeting.

xlogin.Login.greetColor
The color used to display the greeting.

xlogin.Login.namePrompt
The string displayed to prompt for a user name. **Xrdb** strips trailing white space from resource values, so to add spaces at the end of the prompt (usually a nice thing), add spaces escaped with backslashes. The default is ‘‘Login: ’’

xlogin.Login.passwdPrompt
The string displayed to prompt for a password. The default is ‘‘Password: ’’

xlogin.Login.promptFont
The font used to display both prompts.

xlogin.Login.promptColor
The color used to display both prompts.

xlogin.Login.fail

A message which is displayed when the authentication fails. The default is ‘‘Login incorrect’’

xlogin.Login.failFont
The font used to display the failure message.

xlogin.Login.failColor
The color used to display the failure message.

xlogin.Login.failTimeout
The number of seconds that the failure message is displayed. The default is 30.

xlogin.Login.translations
This specifies the translations used for the login widget. Refer to the X Toolkit documentation for a complete discussion on translations. The default translation table is:

```
Ctrl<Key>H:          delete-previous-character() \n\
Ctrl<Key>D:          delete-character() \n\
Ctrl<Key>B:          move-backward-character() \n\
Ctrl<Key>F:          move-forward-character() \n\
Ctrl<Key>A:          move-to-begining() \n\
Ctrl<Key>E:          move-to-end() \n\
Ctrl<Key>K:          erase-to-end-of-line() \n\
Ctrl<Key>U:          erase-line() \n\
Ctrl<Key>X:          erase-line() \n\
Ctrl<Key>C:          restart-session() \n\
Ctrl<Key>\\:         abort-session() \n\
<Key>BackSpace:      delete-previous-character() \n\
<Key>Delete:         delete-previous-character() \n\
<Key>Return:         finish-field() \n\
<Key>:               insert-char() \
```

The actions which are supported by the widget are:

delete-previous-character
Erases the character before the cursor.

delete-character
Erases the character after the cursor.

move-backward-character
Moves the cursor backward.

move-forward-character
Moves the cursor forward.

move-to-begining
(Apologies about the spelling error.) Moves the cursor to the beginning of the editable text.

move-to-end

Moves the cursor to the end of the editable text.

erase-to-end-of-line
: Erases all text after the cursor.

erase-line
: Erases the entire text.

finish-field
: If the cursor is in the name field, proceeds to the password field; if the cursor is in the password field, checks the current name/password pair. If the name/password pair is valid, **xdm** starts the session. Otherwise the failure message is displayed and the user is prompted again.

abort-session
: Terminates and restarts the server.

abort-display
: Terminates the server, disabling it. This is a rash action and is not accessible in the default configuration. It can be used to stop *xdm* when shutting the system down or when using *xdmshell.*

restart-session
: Resets the X server and starts a new session. This can be used when the resources have been changed and you want to test them or when the screen has been overwritten with system messages.

insert-char
: Inserts the character typed.

set-session-argument
: Specifies a single word argument which is passed to the session at startup. See the sections **Session Program** and **Typical Usage**.

allow-all-access
: Disables access control in the server. This can be used when the .Xauthority file cannot be created by **xdm.** Be very careful using this; it might be better to disconnect the machine from the network before doing this.

STARTUP PROGRAM

The *Xstartup* file is typically a shell script. It is run as root and should be very careful about security. This is the place to put commands which add entries to */etc/utmp,* mount users' home directories from file servers, display the message of the day, or abort the session if logins are not allowed. Various environment variables are set for the use of this script:

DISPLAY	the associated display name
HOME	the initial working directory of the user
USER	the user name
PATH	the value of **DisplayManager.DISPLAY.systemPath**
SHELL	the value of **DisplayManager.DISPLAY.systemShell**
XAUTHORITY	may be set to an authority file

No arguments are passed to the script. **Xdm** waits until this script exits before starting the user session. If the exit value of this script is non-zero, **xdm** discontinues the session and starts another authentication cycle.

SESSION PROGRAM

The *Xsession* program is the command which is run as the user's session. It is run with the permissions of the authorized user, and has several environment variables specified:

DISPLAY	the associated display name
HOME	the initial working directory of the user
USER	the user name
PATH	the value of **DisplayManager.DISPLAY.userPath**
SHELL	the user's default shell (from *getpwnam*)
XAUTHORITY	may be set to a non-standard authority file

At most installations, *Xsession* should look in $HOME for a file *.xsession,* which contains commands that each user would like to use as a session. *Xsession* should also implement a system default session if no user-specified session exists. See the section **Typical Usage**.

An argument may be passed to this program from the authentication widget using the 'set-session-argument' action. This can be used to select different styles of session. One good use of this feature is to allow the user to escape from the ordinary session when it fails. This allows users to repair their own *.xsession* if it fails, without requiring administrative intervention. The section **Typical Usage** demonstrates this feature.

RESET PROGRAM

Symmetrical with *Xstartup,* the *Xreset* script is run after the user session has terminated. Run as root, it should contain commands that undo the effects of commands in *Xstartup,* removing entries from */etc/utmp* or unmounting directories from file servers. The environment variables that were passed to *Xstartup* are also passed to *Xreset.*

CONTROLLING THE SERVER

Xdm controls local servers using POSIX signals. SIGHUP is expected to reset the server, closing all client connections and performing other cleanup duties. SIGTERM is expected to terminate the server. If these signals do not perform the expected actions, **xdm** will not perform properly.

To control remote terminals not using XDMCP, **xdm** searches the window hierarchy on the display and uses the protocol request KillClient in an attempt to clean up the terminal for the next session. This may not actually kill all of the clients, as only those which have created windows will be noticed. XDMCP provides a more sure mechanism; when **xdm** closes its initial connection, the session is over and the terminal is required to close all other connections.

CONTROLLING XDM

Xdm responds to two signals: SIGHUP and SIGTERM. When sent a SIGHUP, **xdm** rereads the configuration file, the access control file, and the servers file. For the servers file, it notices if entries have been added or removed. If a new entry has been added, **xdm** starts a session on the associated display. Entries which have been removed are disabled immediately, meaning that any session in progress will be terminated without notice and no new session will be started.

When sent a SIGTERM, **xdm** terminates all sessions in progress and exits. This can be used when shutting down the system.

Xdm attempts to mark its various sub-processes for **ps**(1) by editing the command line argument list in place. Because **xdm** can't allocate additional space for this task, it is useful to start **xdm** with a reasonably long command line (using the full path name should be enough). Each process which is servicing a display is marked *–display.*

OTHER POSSIBILITIES

You can use **xdm** to run a single session at a time, using the 4.3 **init** options or other suitable daemon by specifying the server on the command line:

```
xdm –server ":0 SUN-3 / 60CG4 local  / usr / bin / X :0"
```

Or, you might have a file server and a collection of X terminals. The configuration for this is identical to the sample above, except the *Xservers* file would look like

```
extol:0 VISUAL-19 foreign
exalt:0 NCD-19 foreign
explode:0 NCR-TOWERVIEW3000 foreign
```

This directs **xdm** to manage sessions on all three of these terminals. See the section **Controlling Xdm** for a description of using signals to enable and disable these terminals in a manner reminiscent of **init**(1M).

LIMITATIONS

One thing that **xdm** isn't very good at doing is coexisting with other window systems. To use multiple window systems on the same hardware, you'll probably be more interested in **xinit.**

On the Solaris x86 platform, an attempt to type a login name into **xdm** will cause the keyboard to lock up. The workaround is to use **xdm -nodaemon** instead.

FILES

/usr/openwin/lib/xdm/xdm-config	the default configuration file
/usr/openwin/lib/xdm/Xaccess	the default access file, listing authorized displays
/usr/openwin/lib/xdm/Xservers	the default server file, listing non-XDMCP servers to manage
$(HOME)/.Xauthority	user authorization file where *xdm* stores keys for clients to read
/usr/openwin/lib/xdm/chooser	the default chooser
/usr/openwin/bin/xrdb	the default resource database loader
/usr/openwin/bin/X	the default server
/usr/openwin/bin/xterm	the default session program and failsafe client
/usr/openwin/lib/xdm/A<host>–<suffix>	

the default place for authorization files

SEE ALSO

X11(7), **xinit**(1), **xauth**(1), and XDMCP

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium

NAME

xdpr – dump an X window directly to a printer

SYNOPSIS

xdpr [*filename*] [**-display** *host:display*] [**-P** *printer*] [**-device** *printer_device*] [option...]

DESCRIPTION

xdpr uses the commands **xwd**(1), **xpr**(1), and **lp**(1) to dump an X window, process it for a particular printer type, and print it out on the printer of your choice. This is the easiest way to get a printout of a window. **Xdpr** by default will print the largest possible representation of the window on the output page.

The options for **xdpr** are the same as those for **xpr**, **xwd ,** and **lp .** The most commonly-used options are described below; see the manual pages for these commands for more detailed descriptions of the many options available.

filename
Specifies a file containing a window dump (created by **xwd**) to be printed instead of selecting an X window.

-P *printer*
Specifies a printer to send the output to. If a printer name is not specified here, **xdpr** (really, **lp)** will send your output to the printer specified by the *PRINTER* environment variable. Be sure that type of the printer matches the type specified with the **-device** option.

-display *host:display***[***.screen***]**
Normally, **xdpr** gets the host and display number to use from the environment variable ‘‘DISPLAY’’. One can, however, specify them explicitly; see **X11**(7).

-device *printer-device*
Specifies the device type of the printer. Available printer devices are "ln03" for the DEC LN03, "pp" for the IBM 3812 PagePrinter, and "ps" for any postscript printer (e.g. DEC LN03R or LPS40). The default is "ln03".

-help This option displays the list of options known to **xdpr**.

Any other arguments will be passed to the **xwd**(1), **xpr**(1), and **lp**(1) commands as appropriate for each.

SEE ALSO

xwd(1), **xpr**(1), **lp**(1), **xwud**(1)

ENVIRONMENT

DISPLAY - for which display to use by default.
PRINTER - for which printer to use by default.

COPYRIGHT

Copyright 1985, 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Paul Boutin, MIT Project Athena
Michael R. Gretzinger, MIT Project Athena
Jim Gettys, MIT Project Athena

NAME

xdpyinfo – display information utility for X

SYNOPSIS

xdpyinfo [**-display** *displayname*]

DESCRIPTION

Xdpyinfo is a utility for displaying information about an X server. It is used to examine the capabilities of a server, the predefined values for various parameters used in communicating between clients and the server, and the different types of screens and visuals that are available.

EXAMPLE

The following shows a sample produced by **xdpyinfo** when connected to display that supports an 8 plane screen and a 1 plane screen.

```
name of display:    :0.0
version number:    11.0
vendor string:    MIT X Consortium
vendor release number:    4
maximum request size:  16384 longwords (65536 bytes)
motion buffer size:  0
bitmap unit, bit order, padding:    32, MSBFirst, 32
image byte order:    MSBFirst
number of supported pixmap formats:    2
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
keycode range:    minimum 8, maximum 129
focus:  PointerRoot
number of extensions:    4
    SHAPE
    MIT-SHM
    Multi-Buffering
    MIT-SUNDRY-NONSTANDARD
default screen number:    0
number of screens:    2

screen #0:
  dimensions:    1152x900 pixels (325x254 millimeters)
  resolution:    90x90 dots per inch
  depths (2):    1, 8
  root window id:    0x8006e
  depth of root window:    8 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x8006b
  default number of colormap cells:    256
  preallocated pixels:    black 1, white 0
  options:    backing-store YES, save-unders YES
```

```
current input event mask:    0xd0801d
  KeyPressMask             ButtonPressMask          ButtonReleaseMask
  EnterWindowMask          ExposureMask             SubstructureRedirectMask
  PropertyChangeMask       ColormapChangeMask
number of visuals:    6
default visual id:  0x80065
visual:
  visual id:    0x80065
  class:    PseudoColor
  depth:    8 planes
  size of colormap:    256 entries
  red, green, blue masks:    0x0, 0x0, 0x0
  significant bits in color specification:    8 bits
visual:
  visual id:    0x80066
  class:    DirectColor
  depth:    8 planes
  size of colormap:    8 entries
  red, green, blue masks:    0x7, 0x38, 0xc0
  significant bits in color specification:    8 bits
visual:
  visual id:    0x80067
  class:    GrayScale
  depth:    8 planes
  size of colormap:    256 entries
  red, green, blue masks:    0x0, 0x0, 0x0
  significant bits in color specification:    8 bits
visual:
  visual id:    0x80068
  class:    StaticGray
  depth:    8 planes
  size of colormap:    256 entries
  red, green, blue masks:    0x0, 0x0, 0x0
  significant bits in color specification:    8 bits
visual:
  visual id:    0x80069
  class:    StaticColor
  depth:    8 planes
  size of colormap:    256 entries
  red, green, blue masks:    0x7, 0x38, 0xc0
  significant bits in color specification:    8 bits
visual:
  visual id:    0x8006a
  class:    TrueColor
  depth:    8 planes
```

```
    size of colormap:    8 entries
    red, green, blue masks:    0x7, 0x38, 0xc0
    significant bits in color specification:    8 bits
  number of mono multibuffer types:    6
    visual id, max buffers, depth:    0x80065, 0, 8
    visual id, max buffers, depth:    0x80066, 0, 8
    visual id, max buffers, depth:    0x80067, 0, 8
    visual id, max buffers, depth:    0x80068, 0, 8
    visual id, max buffers, depth:    0x80069, 0, 8
    visual id, max buffers, depth:    0x8006a, 0, 8
  number of stereo multibuffer types:    0

screen #1:
  dimensions:    1152x900 pixels (325x254 millimeters)
  resolution:    90x90 dots per inch
  depths (1):    1
  root window id:    0x80070
  depth of root window:    1 plane
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x8006c
  default number of colormap cells:    2
  preallocated pixels:    black 1, white 0
  options:    backing-store YES, save-unders YES
  current input event mask:    0xd0801d
    KeyPressMask             ButtonPressMask          ButtonReleaseMask
    EnterWindowMask          ExposureMask             SubstructureRedirectMask
    PropertyChangeMask       ColormapChangeMask
  number of visuals:    1
  default visual id:  0x80064
  visual:
    visual id:    0x80064
    class:    StaticGray
    depth:    1 plane
    size of colormap:    2 entries
    red, green, blue masks:    0x0, 0x0, 0x0
    significant bits in color specification:    1 bits
  number of mono multibuffer types:    1
    visual id, max buffers, depth:    0x80064, 0, 1
  number of stereo multibuffer types:    0
```

ENVIRONMENT

DISPLAY
To get the default host, display number, and screen.

SEE ALSO

X11(7), **xwininfo**(1), **xprop**(1), **xrdb**(1)

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xedit – simple text editor for X

SYNOPSIS

xedit [-*toolkitoption* . . .] [*filename*]

DESCRIPTION

Xedit provides a window consisting of the following four areas:

Commands Section	A set of commands that allow you to exit **xedit**, save the file, or load a new file into the edit window.
Message Window	Displays **xedit** messages. In addition, this window can be used as a scratch pad.
Filename Display	Displays the name of the file currently being edited, and whether this file is *Read-Write* or *Read Only*.
Edit Window	Displays the text of the file that you are editing or creating.

OPTIONS

Xedit accepts all of the standard X Toolkit command line options (see **X11**(7)). The order of the command line options is not important.

filename Specifies the file that is to be loaded during start-up. This is the file which will be edited. If a file is not specified, **xedit** lets you load a file or create a new file after it has started up.

EDITING

The Athena Text widget is used for the three sections of this application that allow text input. The characters typed will go to the Text widget that the pointer cursor is currently over. If the pointer cursor is not over a text widget then the keypresses will have no effect on the application. This is also true for the special key sequences that popup dialog widgets, so typing Control-S in the filename widget will enable searching in that widget, not the edit widget.

Both the message window and the edit window will create a scrollbar if the text to display is too large to fit in that window. Horizontal scrolling is not allowed by default, but can be turned on through the Text widget's resources, see *Athena Widget Set* for the exact resource definition.

The following keystroke combinations are defined:

Ctrl-a	Beginning Of Line	Meta-b	Backward Word
Ctrl-b	Backward Character	Meta-f	Forward Word
Ctrl-d	Delete Next Character	Meta-i	Insert File
Ctrl-e	End Of Line	Meta-k	Kill To End Of Paragraph
Ctrl-f	Forward Character	Meta-q	Form Paragraph
Ctrl-g	Multiply Reset	Meta-v	Previous Page
Ctrl-h	Delete Previous Character	Meta-y	Insert Current Selection
Ctrl-j	Newline And Indent	Meta-z	Scroll One Line Down
Ctrl-k	Kill To End Of Line	Meta-d	Delete Next Word
Ctrl-l	Redraw Display	Meta-D	Kill Word
Ctrl-m	Newline	Meta-h	Delete Previous Word
Ctrl-n	Next Line	Meta-H	Backward Kill Word

Ctrl-o	Newline And Backup	Meta-<	Beginning Of File
Ctrl-p	Previous Line	Meta->	End Of File
Ctrl-r	Search/Replace Backward	Meta-]	Forward Paragraph
Ctrl-s	Search/Replace Forward	Meta-[	Backward Paragraph
Ctrl-t	Transpose Characters		
Ctrl-u	Multiply by 4	Meta-Delete	Delete Previous Word
Ctrl-v	Next Page	Meta-Shift Delete	Kill Previous Word
Ctrl-w	Kill Selection	Meta-Backspace	Delete Previous Word
Ctrl-y	Unkill	Meta-Shift Backspace	Kill Previous Word
Ctrl-z	Scroll One Line Up		

In addition, the pointer may be used to cut and paste text:

Button 1 Down	Start Selection
Button 1 Motion	Adjust Selection
Button 1 Up	End Selection (cut)
Button 2 Down	Insert Current Selection (paste)
Button 3 Down	Extend Current Selection
Button 3 Motion	Adjust Selection
Button 3 Up	End Selection (cut)

COMMANDS

Quit Quits the current editing session. If any changes have not been saved, **xedit** displays a warning message, allowing the user to save the file.

Save If file backups are enabled (see RESOURCES, below) **xedit** stores a copy of the original, unedited file in <prefix>*file*<suffix>, then overwrites the *file* with the contents of the edit window. The filename is retrieved from the Text widget directly to the right of the *Load* button.

Load Loads the file named in the text widget immediately to the right of the this button and displays it in the Edit window. If the currently displayed file has been modified a warning message will ask the user to save the changes, or press *Load* again.

RESOURCES

For **xedit** the available resources are:

enableBackups (Class **EnableBackups)**

Specifies that, when edits made to an existing file are saved, **xedit** is to copy the original version of that file to <prefix>*file*<suffix> before it saves the changes. The default value for this resource is ‘‘off,’’ stating that no backups should be created.

backupNamePrefix (Class **BackupNamePrefix)**

Specifies a string that is to be prepended to the backup filename. The default is that no string shall be prepended.

backupNameSuffix (Class **BackupNameSuffix)**

Specifies a string that is to be appended to the backup filename. The default is to use ‘‘.BAK’’ as the suffix.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **xedit**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xedit  xedit
        Paned  paned
                Paned  buttons
                        Command  quit
                        Command  save
                        Command  load
                        Text  filename
                Label  bc_label
                Text  messageWindow
                Label  labelWindow
                Text  editWindow
```

ENVIRONMENT

DISPLAY	to get the default host and display number.
XENVIRONMENT	to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/openwin/lib/app-defaults/Xedit specifies required resources

SEE ALSO

X11(7), **xrdb**(1), *Athena Widget Set*

RESTRICTIONS

There is no *undo* function.

COPYRIGHT

Copyright 1988, Digital Equipment Corporation.
Copyright 1989, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Chris D. Peterson, MIT X Consortium

NAME

xepsf – display an Encapsulated PostScript file

SYNOPSIS

xepsf [**–display** *name*][**–mag** *n*] *filename*

DESCRIPTION

xepsf is a Display PostScript program that displays an Encapsulated PostScript file.

OPTIONS

–display *name*
specifies the display on which to open a connection to the Display PostScript system. If no display is specified, the DISPLAY environment variable is used.

–mag *n* scales the image by a factor of *n*. The scale factor may be either an integer or a floating-point number.

DIAGNOSTICS

Error messages are printed to standard output.

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

NAME

xev – print contents of X events

SYNOPSIS

xev [**-display** *displayname*] [**-geometry** *geom*]

DESCRIPTION

Xev creates a window and then asks the X server to send it notices called *events* whenever anything happens to the window (such as being moved, resized, typed in, clicked in, etc.). It is useful for seeing what causes events to occur and to display the information that they contain.

OPTIONS

-display *display*
This option specifies the X server to contact.

-geometry *geom*
This option specifies the size and / or location of the window.

SEE ALSO

X11(7), **xwininfo**(1), **xdpyinfo**(1),
Xlib Programmers Manual, X Protocol Specification

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xeyes – Eyes follow your pointer

SYNOPSIS

xeyes [-option ...]

DESCRIPTION

xeyes displays a pair of eyes which follow your pointer as it is moved around the display.

OPTIONS

–**fg** *foreground color*
choose a different color for the pupil of the eyes.

–**bg** *background color*
choose a different color for the background.

–**outline** *outline color*
choose a different color for the outline of the eyes.

–**center** *center color*
choose a different color for the center of the eyes.

–**backing WhenMapped | Always | NotUseful**
selects an appropriate level of backing store.

–**geometry** *geometry*
define the initial window geometry; see **X11**(7)**.**

–**display** *display*
specify the display to use; see **X11**(7)**.**

–**bd** *border color*
choose a different color for the window border.

–**bw** *border width*
choose a different width for the window border.

–**shape** uses the SHAPE extension to shape the window.

SEE ALSO

X Toolkit documentation

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Keith Packard, MIT X Consortium

NAME

xfd – display all the characters in an X font

SYNOPSIS

xfd [-*toolkitoptions...*] [**-fn** *fontname*] [**-box**] [**-center**] [**-start** *number*] [**-bc** *color*]

DESCRIPTION

The **xfd** utility creates a window containing the name of the font being displayed, a row of command buttons, several lines of text for displaying character metrics, and a grid containing one glyph per cell. The characters are shown in increasing order from left to right, top to bottom. The first character displayed at the top left will be character number 0 unless the **-start** option has been supplied in which case the character with the number given in the **-start** option will be used.

The characters are displayed in a grid of boxes, each large enough to hold any single character in the font. Each character glyph is drawn using the PolyText16 request (used by the **Xlib** routine **XDrawString16**). If the **-box** option is given, a rectangle will be drawn around each character, showing where an ImageText16 request (used by the **Xlib** routine **XDrawImageString16**) would cause background color to be displayed.

The origin of each glyph is normally set so that the character is drawn in the upper left hand corner of the grid cell. However, if a glyph has a negative left bearing or an unusually large ascent, descent, or right bearing (as is the case with *cursor* font), some character may not appear in their own grid cells. The **-center** option may be used to force all glyphs to be centered in their respective cells.

All the characters in the font may not fit in the window at once. To see the next page of glyphs, press the *Next* button at the top of the window. To see the previous page, press *Prev*. To exit **xfd**, press *Quit.*

Individual character metrics (index, width, bearings, ascent and descent) can be displayed at the top of the window by pressing on the desired character.

The font name displayed at the top of the window is the full name of the font, as determined by the server. See **xlsfonts**(1) for ways to generate lists of fonts, as well as more detailed summaries of their metrics and properties.

OPTIONS

xfd accepts all of the standard toolkit command line options along with the additional options listed below:

-fn *font* This option specifies the font to be displayed.

-box This option indicates that a box should be displayed outlining the area that would be filled with background color by an ImageText request.

-center This option indicates that each glyph should be centered in its grid.

-start *number*
This option specifies the glyph index of the upper left hand corner of the grid. This is used to view characters at arbitrary locations in the font. The default is 0.

-bc *color* This option specifies the color to be used if ImageText boxes are drawn.

SEE ALSO

X11(7), **xlsfonts**(1), **xrdb**(1), **xfontsel**(1)

BUGS

The program should skip over pages full of non-existent characters.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium; previous program of the same name by Mark Lillibridge, MIT Project Athena.

NAME

xfontsel – point & click interface for selecting X11 font names

SYNOPSIS

xfontsel [*-toolkitoption...*] [**-pattern** *fontname*] [**-print**] [**-sample** *text*] [**-sample16** *text16*] [**-noscaled**]

DESCRIPTION

The **xfontsel** application provides a simple way to display the fonts known to your X server, examine samples of each, and retrieve the X Logical Font Description ("XLFD") full name for a font.

If **-pattern** is not specified, all fonts with XLFD 14-part names will be selectable. To work with only a subset of the fonts, specify **-pattern** followed by a partially or fully qualified font name; e.g., ‘‘-pattern *medium*’’ will select that subset of fonts which contain the string ‘‘medium’’ somewhere in their font name. Be careful about escaping wildcard characters in your shell.

If **-print** is specified on the command line the selected font specifier will be written to standard output when the *quit* button is activated. Regardless of whether or not **-print** was specified, the font specifier may be made the PRIMARY (text) selection by activating the *select* button.

The **-sample** option specifies the sample text to be used to display the selected font if the font is linearly indexed, overriding the default.

The **-sample16** option specifies the sample text to be used to display the selected font if the font is matrix encoded, overriding the default.

The **-noscaled** option disables the ability to select scaled fonts at arbitrary pixel or point sizes. This makes it clear which bitmap sizes are advertised by the server, and can avoid an accidental and sometimes prolonged wait for a font to be scaled.

INTERACTIONS

Clicking any pointer button in one of the XLFD field names will pop up a menu of the currently-known possibilities for that field. If previous choices of other fields were made, only values for fonts which matched the previously selected fields will be selectable; to make other values selectable, you must deselect some other field(s) by choosing the ‘‘*’’ entry in that field. Unselectable values may be omitted from the menu entirely as a configuration option; see the **ShowUnselectable** resource, below. Whenever any change is made to a field value, **xfontsel** will assert ownership of the PRIMARY_FONT selection. Other applications (see, e.g., **xterm**(1)) may then retrieve the selected font specification.

Scalable fonts come back from the server with zero for the pixel size, point size, and average width fields. Selecting a font name with a zero in these positions results in an implementation-dependent size. Any pixel or point size can be selected to scale the font to a particular size. Any average width can be selected to anamorphically scale the font (although you may find this challenging given the size of the average width menu).

Clicking the left pointer button in the *select* widget will cause the currently selected font name to become the PRIMARY text selection as well as the PRIMARY_FONT selection. This then allows you to paste the string into other applications. The **select** button remains highlighted to remind you of this fact, and de-highlights when some other application takes the PRIMARY selection away. The *select* widget is a toggle; pressing it when it is highlighted will cause **xfontsel** to release the selection ownership and de-highlight the widget. Activating the *select* widget twice is the only way to cause **xfontsel** to release the PRIMARY_FONT selection.

RESOURCES

The application class is **XFontSel**. Most of the user-interface is configured in the app-defaults file; if this file is missing a warning message will be printed to standard output and the resulting window will be nearly incomprehensible.

Most of the significant parts of the widget hierarchy are documented in the app-defaults file (normally /usr/openwin/lib/app-defaults/XFontSel).

Application specific resources:

cursor (class **Cursor)**
Specifies the cursor for the application window.

pattern (class **Pattern)**
Specifies the font name pattern for selecting a subset of available fonts. Equivalent to the **-pattern** option. Most useful patterns will contain at least one field delimiter; e.g. ‘‘*-m-*’’ for monospaced fonts.

pixelSizeList (class **PixelSizeList)**
Specifies a list of pixel sizes to add to the pixel size menu, so that scalable fonts can be selected at those pixel sizes. The default pixelSizeList contains 7, 30, 40, 50, and 60.

pointSizeList (class **PointSizeList)**
Specifies a list of point sizes (in units of tenths of points) to add to the point size menu, so that scalable fonts can be selected at those point sizes. The default pointSizeList contains 250, 300, 350, and 400.

printOnQuit (class **PrintOnQuit)**
If *True* the currently selected font name is printed to standard output when the quit button is activated. Equivalent to the **-print** option.

sampleText (class **Text)**
The sample 1-byte text to use for linearly indexed fonts. Each glyph index is a single byte, with newline separating lines.

sampleText16 (class **Text16)**
The sample 2-byte text to use for matrix-encoded fonts. Each glyph index is two bytes, with a 1-byte newline separating lines.

scaledFonts (class **ScaledFonts)**
If *True* then selection of arbitrary pixel and point sizes for scalable fonts is enabled.

Widget specific resources:

showUnselectable (class **ShowUnselectable)**
Specifies, for each field menu, whether or not to show values that are not currently selectable, based upon previous field selections. If shown, the unselectable values are clearly identified as such and do not highlight when the pointer is moved down the menu. The full name of this resource is

fieldN.menu.options.showUnselectable, class **MenuButton.SimpleMenu.Options.ShowUnselectable**; where N is replaced with the field number (starting with the left-most field numbered 0). The default is True for all but field 11 (average width of characters in font) and False for field 11. If you never want to see unselectable entries, '*menu.options.showUnselectable:False' is a reasonable thing to specify in a resource file.

FILES

$XFILESEARCHPATH / XFontSel

SEE ALSO

xrdb(1), **xfd**(1)

BUGS

Sufficiently ambiguous patterns can be misinterpreted and lead to an initial selection string which may not correspond to what the user intended and which may cause the initial sample text output to fail to match the proffered string. Selecting any new field value will correct the sample output, though possibly resulting in no matching font.

Should be able to return a FONT for the PRIMARY selection, not just a STRING.

Any change in a field value will cause **xfontsel** to assert ownership of the PRIMARY_FONT selection. Perhaps this should be parameterized.

When running on a slow machine, it is possible for the user to request a field menu before the font names have been completely parsed. An error message indicating a missing menu is printed to stderr but otherwise nothing bad (or good) happens.

The average-width menu is too large to be useful.

COPYRIGHT

AUTHOR

Ralph R. Swick, Digital Equipment Corporation / MIT Project Athena

NAME

xgc – X graphics demo

SYNOPSIS

xgc [-toolkitoption]

DESCRIPTION

The **xgc** program demonstrates various features of the X graphics primitives. In X, most of the details about the graphics to be generated are stored in a resource called a graphics context (GC). The **xgc** program provides a user interface for setting various GC components. Pressing the "Run" button causes these results to be displayed in the large drawing window on the right. Timing information is displayed in the window immediately below.

The items in the upper left hand window work as follows:

Function – specify the logical function with which primitives will drawn. The most usual setting is "set", i.e. simply to render pixels without regard to what has been drawn before.

LineStyle – specify whether lines should be drawn solid in foreground, dashed in foreground or alternating foreground and background.

CapStyle – specify the appearance of the ends of a line.

JoinStyle – specify the appearance of joints between consecutive lines drawn within a single graphics primitive.

solid, tiled with a pixmap or stippled.

rule means that if areas overlap an odd number of times, they are not drawn. Winding rule means that overlapping areas are always filled, regardless of how many times they overlap.

ArcMode – specifies the rule for filling of arcs. The boundary of the arc is either a Chord or two radii.

planemask – specifies which planes of the drawing window are modified. By default, all planes are modified.

dashlist – specifies a pattern to be used when drawing dashed lines.

Line Width – specifies the width in pixels of lines to be drawn. Zero means to draw using the server's fastest algorithm with a line width of one pixel.

Font – specifies the font to be used for text primitives.

Foreground and *Background* – specify the pixel values to be applied when drawing primitives. The Foreground value is used as the pixel value for set bits in the source in all primitives. The Background value is used as the pixel value for unset bits in the source when using Copy Plane, drawing lines with LineStyle of DoubleDash and filling with FillStyle of OpaqueStippled.

Percentage of Test – scrollbar permits specifying only a percentage of the test to be run. The number at the left indicates the current setting, which defaults to 100%.

The window labeled "Test" permits choice of one a number of graphics primitive tests, including Points, Segments, Lines, Arcs and Filled Arcs, 8-bit Text and Image Text, Rectangles and Filled Rectangles, Image draws, as well as Copy Plane and Copy Area.

The window to the right of this has buttons which permit record/playback of the primitives rendered.

OPTIONS

Xgc accepts all of the standard X Toolkit command line options.

X DEFAULTS

This program accepts the usual defaults for toolkit applications.

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X11(7),
O'Reilly Xlib Programming Manual, Volume I, Chapter 5

BUGS

This program isn't really finished yet. In particular, buttons whose labels appear in parentheses indicate features which are not yet implemented.

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.

AUTHORS

Dan Schmidt, MIT

NAME

xhost – server access control program for X

SYNOPSIS

xhost [[+-]name ...]

DESCRIPTION

The **xhost** program is used to add and delete host names or user names to the list allowed to make connections to the X server. In the case of hosts, this provides a rudimentary form of privacy control and security. It is only sufficient for a workstation (single user) environment, although it does limit the worst abuses. Environments which require more sophisticated measures should implement the user-based mechanism, or use the hooks in the protocol for passing other authentication data to the server.

Hostnames that are followed by two colons (::) are used in checking DECnet connections; all other hostnames are used for TCP/IP connections.

User names contain an at-sign (@). When Secure RPC is being used, the network independent netname (e.g., "unix.*uid@domainname*") can be specified, or a local user can be specified with just the username and a trailing at-sign (e.g., "joe@").

OPTIONS

Xhost accepts the following command line options described below. For security, the options that effect access control may only be run from the "controlling host". For workstations, this is the same machine as the server. For X terminals, it is the login host.

[+]*name* The given *name* (the plus sign is optional) is added to the list allowed to connect to the X server. The name can be a host name or a user name.

–*name* The given *name* is removed from the list of allowed to connect to the server. The name can be a host name or a user name. Existing connections are not broken, but new connection attempts will be denied. Note that the current machine is allowed to be removed; however, further connections (including attempts to add it back) will not be permitted. Resetting the server (thereby breaking all connections) is the only way to allow local connections again.

+ Access is granted to everyone, even if they aren't on the list (i.e., access control is turned off).

– Access is restricted to only those on the list (i.e., access control is turned on).

nothing If no command line arguments are given, a message indicating whether or not access control is currently enabled is printed, followed by the list of those allowed to connect. This is the only option that may be used from machines other than the controlling host.

DIAGNOSTICS

For each name added to the access control list, a line of the form "*name* being added to access contro list" is printed. For each name removed from the access control list, a line of the form "*name* being removed from access contro list" is printed.

FILES

/etc/X*.hosts

SEE ALSO

X11(7), **Xserver**(1), **xdm**(1)

ENVIRONMENT

DISPLAY
to get the default host and display to use.

BUGS

You can't specify a display on the command line because **−display** is a valid command line argument (indicating that you want to remove the machine named ''*display*'' from the access list).

This is not really a bug, but the X server stores network addresses, not host names. If somehow you change a host's network address while the server is still running, *xhost* must be used to add the new address and / or remove the old address.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11(7)** for a full statement of rights and permissions.

AUTHORS

Bob Scheifler, MIT Laboratory for Computer Science,
Jim Gettys, MIT Project Athena (DEC).

NAME

xinit – X Window System initializer

SYNOPSIS

xinit [[*client*] *options*] [–– [*server*] [*display*] *options*]

DESCRIPTION

The **xinit** program is used to start the X Window System server and a first client program on systems that cannot start X directly from */etc/init* or in environments that use multiple window systems. When this first client exits, **xinit** will kill the X server and then terminate.

If no specific client program is given on the command line, **xinit** will look for a file in the user's home directory called *.xinitrc* to run as a shell script to start up client programs. If no such file exists, **xinit** will use the following as a default:

```
xterm  –geometry  +1+1  –n  login  –display  :0
```

If no specific server program is given on the command line, **xinit** will look for a file in the user's home directory called *.xserverrc* to run as a shell script to start up the server. If no such file exists, **xinit** will use the following as a default:

```
X  :0
```

Note that this assumes that there is a program named *X* in the current search path. However, servers are usually named *Xdisplaytype* where *displaytype* is the type of graphics display which is driven by this server. The site administrator should, therefore, make a link to the appropriate type of server on the machine, or create a shell script that runs **xinit** with the appropriate server.

An important point is that programs which are run by *.xinitrc* should be run in the background if they do not exit right away, so that they don't prevent other programs from starting up. However, the last long-lived program started (usually a window manager or terminal emulator) should be left in the foreground so that the script won't exit (which indicates that the user is done and that **xinit** should exit).

An alternate client and / or server may be specified on the command line. The desired client program and its arguments should be given as the first command line arguments to **xinit**. To specify a particular server command line, append a double dash (––) to the **xinit** command line (after any client and arguments) followed by the desired server command.

Both the client program name and the server program name must begin with a slash (/) or a period (.). Otherwise, they are treated as an arguments to be appended to their respective startup lines. This makes it possible to add arguments (for example, foreground and background colors) without having to retype the whole command line.

If an explicit server name is not given and the first argument following the double dash (––) is a colon followed by a digit, **xinit** will use that number as the display number instead of zero. All remaining arguments are appended to the server command line.

EXAMPLES

Below are several examples of how command line arguments in **xinit** are used.

xinit
This will start up a server named *X* and run the user's *.xinitrc*, if it exists, or else start an *xterm*.

xinit –– /usr/bin/X11/Xqdss :1
This is how one could start a specific type of server on an alternate display.

xinit –geometry =80x65+10+10 –fn 8x13 –j –fg white –bg navy
This will start up a server named *X*, and will append the given arguments to the default *xterm* command. It will ignore *.xinitrc*.

xinit –e widgets –– ./Xsun –l –c
This will use the command *./Xsun –l –c* to start the server and will append the arguments *–e widgets* to the default *xterm* command.

xinit /usr/ucb/rsh fasthost cpupig –display ws:1 –– :1 –a 2 –t 5
This will start a server named *X* on display 1 with the arguments *–a 2 –t 5*. It will then start a remote shell on the machine **fasthost** in which it will run the command *cpupig*, telling it to display back on the local workstation.

Below is a sample *.xinitrc* that starts a clock, several terminals, and leaves the window manager running as the ''last'' application. Assuming that the window manager has been configured properly, the user then chooses the ''Exit'' menu item to shut down X.

```
xrdb –load $HOME/.Xresources
xsetroot –solid gray &
xclock –g 50x50–0+0 –bw 0 &
xload –g 50x50–50+0 –bw 0 &
xterm –g 80x24+0+0 &
xterm –g 80x24+0–0 &
twm
```

Sites that want to create a common startup environment could simply create a default *.xinitrc* that references a site-wide startup file:

```
#!/bin/sh
. /usr/local/lib/site.xinitrc
```

Another approach is to write a script that starts **xinit** with a specific shell script. Such scripts are usually named *x11*, *xstart*, or *startx* and are a convenient way to provide a simple interface for novice users:

```
#!/bin/sh
xinit /usr/local/lib/site.xinitrc –– /usr/bin/X11/X bc
```

ENVIRONMENT VARIABLES

DISPLAY This variable gets set to the name of the display to which clients should connect.

XINITRC This variable specifies an init file containing shell commands to start up the initial windows. By default, *.xinitrc* in the home directory will be used.

FILES

.xinitrc	default client script
xterm	client to run if *.xinitrc* does not exist
.xserverrc	default server script
X	server to run if *.xserverrc* does not exist

SEE ALSO

olwm(1), **openwin**(1), **props**(1), **X11**(7), **Xserver**(1), **xterm**(1)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science

NAME

xkill – kill a client by its X resource

SYNOPSIS

xkill [–display *displayname*] [–id *resource*] [–button number] [–frame] [–all]

DESCRIPTION

Xkill is a utility for forcing the X server to close connections to clients. This program is very dangerous, but is useful for aborting programs that have displayed undesired windows on a user's screen. If no resource identifier is given with *-id*, **xkill** will display a special cursor as a prompt for the user to select a window to be killed. If a pointer button is pressed over a non-root window, the server will close its connection to the client that created the window.

OPTIONS

–**display** *displayname*
This option specifies the name of the X server to contact.

–**id** *resource*
This option specifies the X identifier for the resource whose creator is to be aborted. If no resource is specified, **xkill** will display a special cursor with which you should select a window to be kill.

–**button** *number*
This option specifies the number of pointer button that should be used in selecting a window to kill. If the word "any" is specified, any button on the pointer may be used. By default, the first button in the pointer map (which is usually the leftmost button) is used.

–**all** This option indicates that all clients with top-level windows on the screen should be killed. **Xkill** will ask you to select the root window with each of the currently defined buttons to give you several chances to abort. Use of this option is highly discouraged.

–**frame** This option indicates that **xkill** should ignore the standard conventions for finding top-level client windows (which are typically nested inside a window manager window), and simply believe that you want to kill direct children of the root.

XDEFAULTS

Button Specifies a specific pointer button number or the word "any" to use when selecting windows.

SEE ALSO

X11(7), **xwininfo**(1), XKillClient and XGetPointerMapping in the Xlib Programmers Manual, KillClient in the X Protocol Specification

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11(7)** for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium
Dana Chee, Bellcore

NAME

xload – system load average display for X

SYNOPSIS

xload [*toolkitoptions*] [**-scale** *integer*] [**-update** *seconds*] [**-hl** *color*] [**-highlight** *color*] [**-jumpscroll** *pixels*] [**-label** *string*] [**-nolabel**] [**-lights**]

DESCRIPTION

The **xload** program displays a periodically updating histogram of the system load average.

OPTIONS

Xload accepts all of the standard X Toolkit command line options (see **X11**(7)). The order of the options in unimportant. **Xload** also accepts the following additional options:

–**hl** *color* or –**highlight** *color*
: This option specifies the color of the scale lines.

–**jumpscroll** *pixels*
: The number of pixels to shift the graph to the left when the graph reaches the right edge of the window. The default value is 1/2 the width of the current window. Smooth scrolling can be achieved by setting it to 1.

–**label** *string*
: The string to put into the label above the load average.

–**nolabel**
: If this command line option is specified then no label will be displayed above the load graph.

–**lights**
: When specified, this option causes **xload** to display the current load average by using the keyboard leds; for a load average of *n*, **xload** lights the first *n* keyboard leds. This option turns off the usual screen display.

–**scale** *integer*
: This option specifies the minimum number of tick marks in the histogram, where one division represents one load average point. If the load goes above this number, **xload** will create more divisions, but it will never use fewer than this number. The default is 1.

–**update** *seconds*
: This option specifies the interval in seconds at which **xload** updates its display. The minimum amount of time allowed between updates is 1 second. The default is 10.

RESOURCES

In addition to the resources available to each of the widgets used by **xload** there is one resource defined by the application itself.

showLabel (class **Boolean)**
: If False then no label will be displayed.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **xload**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XLoad  xload
        Paned  paned
                Label  label
                StripChart  load
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/openwin/lib/app-defaults/XLoad - specifies required resources

SEE ALSO

X11(7), **xrdb**(1), **mem**(7), Athena StripChart Widget.

BUGS

This program requires the ability to open and read the special system file */dev/kmem*. Sites that do not allow general access to this file should make **xload** belong to the same group as */dev/kmem* and turn on the *set group id* permission flag.

Reading */dev/kmem* is inherently non-portable. Therefore, the routine used to read it (get_load.c) must be ported to each new operating system.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHORS

K. Shane Hartman (MIT-LCS) and Stuart A. Malone (MIT-LCS);
with features added by Jim Gettys (MIT-Athena), Bob Scheifler (MIT-LCS), Tony Della Fera (MIT-Athena), and Chris Peterson (MIT-LCS).

NAME

xlock – locks the local X display until a password is entered

SYNOPSIS

xlock [–**display** *dsp*] [–**help**] [–**name** *resource-name*] [–**resources**] [**-/+remote**] [**-/+mono**] [**-/+nolock**] [**-/+allowroot**] [**-/+enablesaver**] [**-/+allowaccess**] [**-/+echokeys**] [**-/+usefirst**] [**-/+v**] [–**delay** *usecs*] [–**batchcount** *num*] [–**nice** *level*] [–**timeout** *seconds*] [–**saturation** *value*] [–**font** *fontname*] [–**bg** *color*] [–**fg** *color*] [–**mode** *modename*] [–**username** *textstring*] [–**password** *textstring*] [–**info** *textstring*] [–**validate** *textstring*] [–**invalid** *textstring*]

DESCRIPTION

xlock locks the X server till the user enters their password at the keyboard. While **xlock** is running, all new server connections are refused. The screen saver is disabled. The mouse cursor is turned off. The screen is blanked and a changing pattern is put on the screen. If a key or a mouse button is pressed then the user is prompted for the password of the user who started **xlock**.

If the correct password is typed, then the screen is unlocked and the X server is restored. When typing the password Control-U and Control-H are active as kill and erase respectively. To return to the locked screen, click in the small icon version of the changing pattern.

OPTIONS

–**display** *dsp*

The *display* option sets the X11 display to lock. **xlock** locks all available screens on a given server, and restricts you to locking only a local server such as **unix:0, localhost:0,** or **:0** unless you set the –**remote** option.

–**name** *resource-name*

resource-name is used instead of **XLock** when looking for resources to configure **xlock**.

–**mode** *modename*

As of this writing there are eight display modes supported (plus one more for random selection of one of the eight).

hop	Hop mode shows the "real plane fractals" from the September 1986 issue of Scientific American.
life	Life mode shows Conway's game of life.
qix	Qix mode shows the spinning lines similar to the old video game by the same name.
image	Image mode shows several sun logos randomly appearing on the screen.
swarm	Swarm mode shows a swarm of bees following a wasp.
rotor	Rotor mode shows a swirling rotorlike thing.
pyro	Pyro mode shows fireworks.

flame Flame mode shows wierd but cool fractals.

blank Blank mode shows nothing but a black screen.

random Random mode picks a random mode from all of the above except blank mode.

–**delay** *usecs*
The *delay* option sets the speed at which a mode will operate. It simply sets the number of microseconds to delay between batches of animations. In blank mode, it is important to set this to some small number of seconds, because the keyboard and mouse are only checked after each delay, so you cannot set the delay too high, but a delay of zero would needlessly consume cpu checking for mouse and keyboard input in a tight loop, since blank mode has no work to do.

–**batchcount** *num*
The *batchcount* option sets number of *things* to do per batch to *num* . In hop mode this refers to the number of pixels rendered in the same color. In life mode it is the number of generations to let each species live. In qix mode it is the number of lines rendered in the same color. In image mode it is the number of sunlogos on screen at once. In swarm mode it is the number of bees. In rotor mode it is the number of rotor thingys which whirr... In pyro mode it is the maximum number flying rockets at one time. In flame mode it is the number of levels to recurse (larger = more complex). In blank mode it means nothing.

–**nice** *nicelevel*
The *nice* option sets system nicelevel of the **xlock** process to *nicelevel* .

–**timeout** *seconds*
The *timeout* option sets the number of *seconds* before the password screen will time out.

–**saturation** *value*
The *saturation* option sets saturation of the color ramp used to *value* . 0 is grayscale and 1 is very rich color. 0.4 is a nice pastel.

–**font** *fontname*
The *font* option sets the font to be used on the prompt screen.

–**fg** *color*
The *fg* option sets the color of the text on the password screen to *color* .

–**bg** *color*
The *bg* option sets the color of the background on the password screen to *color* .

–**username** *textstring*
textstring is shown in front of user name, defaults to "Name: ".

–**password** *textstring*
textstring is the password prompt string, defaults to "Password: ".

–**info** *textstring*
textstring is an informational message to tell the user what to do, defaults to "Enter

password to unlock; select icon to lock.".

–validate *textstring*
textstring –**validate** *message shown while validating the password, defaults to* "Validating login..."

–invalid *textstring*
textstring –**invalid** *message shown when password is invalid, defaults to* "Invalid login."

–resources
The *resources* option prints the default resource file for **xlock** to standard output.

-/+remote
The *remote* option tells **xlock** to not stop you from locking remote X11 servers. This option should be used with care and is intended mainly to lock X11 terminals which cannot run **xlock** locally. If you lock someone else's workstation, they will have to know **your** password to unlock it. Using *+remote* overrides any resource derived values for *remote* and prevents **xlock** from being used to lock other X11 servers. (Use '+' instead of '-' to override resources for other options that can take the '+' modifier similarly.)

-/+mono
The *mono* option causes **xlock** to display monochrome, (black and white) pixels rather than the default colored ones on color displays.

+/-nolock
The *nolock* option causes **xlock** to only draw the patterns and not lock the display. A keypress or a mouse click will terminate the screen saver.

-/+allowroot
The *allowroot* option allows the root password to unlock the server as well as the user who started **xlock**.

-/+enablesaver
By default **xlock** will disable the normal X server's screen saver since it is in effect a replacement for it. Since it is possible to set delay parameters long enough to cause phosphor burn on some displays, this option will turn back on the default screensaver which is very careful to keep most of the screen black.

-/+allowaccess
This option is required for servers which do not allow clients to modify the host access control list. It is also useful if you need to run x clients on a server which is locked for some reason... When allowaccess is true, the X11 server is left open for clients to attach and thus lowers the inherent security of this lockscreen. A side effect of using this option is that if **xlock** is killed -KILL, the access control list is not lost.

-/+echokeys
The *echokeys* option causes **xlock** to echo '?' characters for each key typed into the password prompt. Some consider this a security risk, so the default is to not echo anything.

-/+usefirst

The *usefirst* option causes **xlock** to use the keystroke which got you to the password screen as the first character in the password. The default is to ignore the first key pressed.

–v Verbose mode, tells what options it is going to use.

WARNING

xlock can appear to hang if it is competing with a high-priority process for the CPU. For example, if **xlock** is started after a process with 'nice -20' (high priority), **xlock** will take considerable amount of time to respond.

BUGS

"kill -KILL **xlock** " causes the server that was locked to be unusable, since all hosts (including localhost) were removed from the access control list to lock out new X clients, and since **xlock** couldn't catch SIGKILL, it terminated before restoring the access control list. This will leave the X server in a state where *"you can no longer connect to that server, and this operation cannot be reversed unless you reset the server."*

-From the X11R4 Xlib Documentation, Chapter 7.

SEE ALSO

Xlib Documentation.

AUTHOR

Patrick J. Naughton

COPYRIGHT

NAME

xlogo – X Window System logo

SYNOPSIS

xlogo [-*toolkitoption* ...]

DESCRIPTION

The **xlogo** program displays the X Window System logo. This program is nothing more than a wrapper around the *undocumented* Athena Logo widget.

OPTIONS

Xlogo accepts all of the standard X Toolkit command line options, as well as the following:

–**shape** This option indicates that the logo window should be shaped rather than rectangular.

RESOURCES

The default width and the default height are each 100 pixels. This program uses the *Logo* widget in the Athena widget set. It understands all of the Simple widget resource names and classes as well as:

foreground (class **Foreground)**
Specifies the color for the logo. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *XtDefaultForeground*, otherwise the default is *XtDefaultBackground*.

shapeWindow (class **ShapeWindow)**
Specifies that the window is shaped to the X logo. The default is False.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **xlogo.** In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XLogo  xlogo
        Logo  xlogo
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/openwin/lib/app-defaults/XLogo - specifies required resources

SEE ALSO

X11(7), **xrdb**(1)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11(7)** for a full statement of rights and permissions.

AUTHORS

Ollie Jones of Apollo Computer and Jim Fulton of the MIT X Consortium wrote the logo graphics routine, based on a graphic design by Danny Chong and Ross Chapman of Apollo Computer.

NAME

xlsatoms – list interned atoms defined on server

SYNOPSIS

xlsatoms [-options ...]

DESCRIPTION

Xlsatoms lists the interned atoms. By default, all atoms starting from 1 (the lowest atom value defined by the protocol) are listed until unknown atom is found. If an explicit range is given, **xlsatoms** will try all atoms in the range, regardless of whether or not any are undefined.

OPTIONS

–**display** *dpy*
This option specifies the X server to which to connect.

–**format** *string*
This option specifies a *printf*-style string used to list each atom <*value,name*> pair, printed in that order (*value* is an *unsigned long* and *name* is a *char* *). **Xlsatoms** will supply a newline at the end of each line. The default is *%ld\t%s*.

–**range** *[low]-[high]*
This option specifies the range of atom values to check. If *low* is not given, a value of 1 assumed. If *high* is not given, *xlsatoms* will stop at the first undefined atom at or above *low*.

–**name** *string*
This option specifies the name of an atom to list. If the atom does not exist, a message will be printed on the standard error.

SEE ALSO

X11(7), **Xserver**(1), **xprop**(1)

ENVIRONMENT

DISPLAY
to get the default host and display to use.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xlsclients – list client applications running on a display

SYNOPSIS

xlsclients [-display *displayname*] [-a] [-l] [-m maxcmdlen]

DESCRIPTION

Xlsclients is a utility for listing information about the client applications running on a display. It may be used to generate scripts representing a snapshot of the the user's current session.

OPTIONS

–display *displayname*
This option specifies the X server to contact.

–a This option indicates that clients on all screens should be listed. By default, only those clients on the default screen are listed.

–l This option indicates that a long listing showing the window name, icon name, and class hints in addition to the machine name and command string shown in the default listing.

–m *maxcmdlen*
This option specifies the maximum number of characters in a command to print out. The default is 10000.

ENVIRONMENT

DISPLAY
To get the default host, display number, and screen.

SEE ALSO

X11(7), **xwininfo**(1), **xprop**(1)

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.
See **X11(7)** for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xlsfonts – server font list displayer for X

SYNOPSIS

xlsfonts [-options ...] [-fn pattern]

DESCRIPTION

Xlsfonts lists the fonts that match the given *pattern*. The wildcard character "*" may be used to match any sequence of characters (including none), and "?" to match any single character. If no pattern is given, "*" is assumed.

The "*" and "?" characters must be quoted to prevent them from being expanded by the shell.

OPTIONS

–display *host*:*dpy*
: This option specifies the X server to contact.

–l[l[l]] This option indicates that medium, long, and very long listings, respectively, should be generated for each font.

–m This option indicates that long listings should also print the minimum and maximum bounds of each font.

–C This option indicates that listings should use multiple columns. This is the same as **-n 0**.

–1 This option indicates that listings should use a single column. This is the same as **-n 1**.

–w *width*
: This option specifies the width in characters that should be used in figuring out how many columns to print. The default is 79.

–n *columns*
: This option specifies the number of columns to use in displaying the output. By default, it will attempt to fit as many columns of font names into the number of character specified by **-w** *width*.

–u This option indicates that the output should be left unsorted.

–o This option indicates that **xlsfonts** should do an **OpenFont** (and **QueryFont**, if appropriate) rather than a **ListFonts**. This is useful if **ListFonts** or **ListFontsWithInfo** fail to list a known font (as is the case with some scaled font systems).

SEE ALSO

X11(7), **Xserver**(1), **xset**(1), **xfd**(1)

ENVIRONMENT

DISPLAY
: to get the default host and display to use.

BUGS

Doing ‘‘xlsfonts -l’’ can tie up your server for a very long time. This is really a bug with single-threaded non-preemptable servers, not with this program.

COPYRIGHT

AUTHOR

Mark Lillibridge, MIT Project Athena; Jim Fulton, MIT X Consortium; Phil Karlton, SGI

NAME

xlswins – server window list displayer for X

SYNOPSIS

xlswins [-options ...] [*windowid* ...]

DESCRIPTION

xlswins lists the window tree. By default, the root window is used as the starting point, although a specific window may be specified using the *-id* option. If no specific windows are given on the command line, the root window will be used.

OPTIONS

–**display** *displayname*
This option specifies the X server to contact.

–**l** This option indicates that a long listing should be generated for each window. This includes a number indicating the depth, the geometry relative to the parent as well as the location relative to the root window.

–**format** *radix*
This option specifies the radix to use when printing out window ids. Allowable values are: *hex*, *octal*, and *decimal*. The default is hex.

–**indent** *number*
This option specifies the number of spaces that should be indented for each level in the window tree. The default is 2.

SEE ALSO

xprop(1), **xwininfo**(1)

ENVIRONMENT

DISPLAY
to get the default host and display to use.

BUGS

This should be integrated with xwininfo somehow.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xmac – display Apple MacPaint image files under X windows

SYNOPSIS

xmac *filename* [**-ps**] [*host:display*] [*-geometry*]

DESCRIPTION

xmac displays a MacPaint file in a window, allows resize / move, and has an icon. **xmac** will send the Postscript commands to print the image to standard out if you include the command line option, **-ps. xmac** accepts two other optional command line arguments. You may specify a display name in the form *host:display* (see **X11**(7). And you may provide a geometry specification. If you don't give a geometry specification, **xmac** will ask you where you want to put the window when it starts up. See **X11**(7) for a full explanation.

BUGS

There are no known bugs. There are lots of lacking features. I would like to add editing capability in the future, along with the ability to clip part of an image to a bitmap format file; as well as replacing the desktop pattern with an image. Also there should be a way to kill the process, i.e. a keypress or a mouse click in a box. Also a title bar would be nice.

ENVIRONMENT

XMAC - the default directory for searching for image files, (after ".").

SEE ALSO

X11(7),
Xlib Documentation.

AUTHOR

Copyright (c) 1987 by Patrick J. Naughton, (naughton@sun.soe.clarkson.edu)

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

NAME

xmag – magnify parts of the screen

SYNOPSIS

xmag [**–mag** *magfactor*] [**–source** *geom*] [*–toolkitoption* ...]

DESCRIPTION

The **xmag** program allows you to magnify portions of an X screen. If no explicit region is specified, a square with the pointer in the upper left corner is displayed indicating the area to be enlarged. The area can be dragged out to the desired size by pressing Button 2. Once a region has been selected, a window is popped up showing a blown up version of the region in which each pixel in the source image is represented by a small square of the same color. Pressing Button1 in the enlargement window shows the position and RGB value of the pixel under the pointer until the button is released. Typing ‘‘Q’’ or ‘‘ˆC’’ in the enlargement window exits the program. The application has 5 buttons across its top. *Close* deletes this particular magnification instance. *Replace* brings up the rubber band selector again to select another region for this magnification instance. *New* brings up the rubber band selector to create a new magnification instance. *Cut* puts the magnification image into the primary selection. *Paste* copies the primary selection buffer into **xmag.** Note that you can cut and paste between **xmag** and the **bitmap**(1) program. Resizing **xmag** resizes the magnification area. **xmag** preserves the colormap, visual, and window depth of the source.

WIDGETS

xmag uses the X Toolkit and the Athena Widget Set. The magnified image is displayed in the Scale widget. For more information, see the Athena Widget Set documentation. Below is the widget structure of the **xmag** application. Indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xmag xmag
        RootWindow root
        TopLevelShell xmag
                Paned pane1
                        Paned pane2
                                Command close
                                Command replace
                                Command new
                                Command select
                                Command paste
                                Label xmag label
                        Paned pane2
                                Scale scale
        OverrideShell pixShell
                Label pixLabel
```

OPTIONS

–source *geom* This option specifies the size and / or location of the source region on the screen. By default, a 64x64 square is provided for the user to select an area of the screen.

–mag *integer* This option indicates the magnification to be used. 5 is the default.

COPYRIGHT

Copyright 1991, Massachusetts Institute of Technology.
See **X11(7),** for a full statement of rights and permissions.

SEE ALSO

X11(7), **xmag_multivis**(6)

AUTHORS

Dave Sternlicht and Davor Matic, MIT X Consortium.

NAME

xmag_multivis – magnify parts of the screen

SYNOPSIS

xmag_multivis [–**display** *display*] [–**geometry** *geom*] [–**source** *geom*] [–**mag** *magfactor*] [–**bw** *pixels*] [–**bd** *color*] [–**bg** *colorpixelvalue*] [–**fn** *fontname*] [–**z**]

DESCRIPTION

The **xmag_multivis** program allows you to magnify portions of the screen. If no explicit region is specified, a square centered around the pointer is displayed indicating the area to be enlarged. Once a region has been selected, a window is popped up showing a blown up version of the region in which each pixel in the source image is represented by a small square of the same color. Pressing Button1 on the pointer in the enlargement window pops up a small window displaying, in hexadecimal notation, the position, pixel value, and RGB value of the pixel under the pointer until the button is released. Pressing the space bar or any other pointer button removes the enlarged image so that another region may be selected. Pressing ‘‘q’’, ‘‘Q’’, or ‘‘ˆC’’ in the enlargement window exits the program.

On displays that export multiple Visuals, the selected region is first examined for overlaps by windows of differing depth, visual or colormap. If the selected region does contain such windows, the image displayed in the enlargement window is constructed by compositing the images from the various windows under the selected region. In this case, the composite image will only be displayed in a TrueColor 24 bit deep window, provided such a window can be created on that screen. The image displayed in the enlargement window will exhibit true colors for every pixel in the composite image, regardless of the Colormap state (Installed / UnInstalled) of any of the windows within the selected region.

OPTIONS

–**display** *display*
: This option specifies the X server to use for both reading the screen and displaying the enlarged version of the image.

–**geometry** *geom*
: This option specifies the size and / or location of the enlargement window. By default, the size is computed from the size of the source region and the desired magnification. Therefore, only one of –**source** *size* and –**mag** *magfactor* options may be specified if a window size is given with this option.

–**source** *geom*
: This option specifies the size and / or location of the source region on the screen. By default, a 64x64 square centered about the pointer is provided for the user to select an area of the screen. The size of the source is used with the desired magnification to compute the default enlargement window size. Therefore, only one of –**geometry** *size* and –**mag** *magfactor* options may be specified if a source size is given with this option.

–**mag** *magfactor*
: This option specifies an integral factor by which the source region should be enlarged. The default magnification is 5. This is used with the size of the source to compute the default enlargement window size. Therefore, only one of

–**geometry** *size* and –**source** *geom* options may be specified if a magnification factor is given with this option.

–**bw** *pixels*
: This option specifies the width in pixels of the border surrounding the enlargement window.

–**bd** *color*
: This option specifies the color to use for the border surrounding the enlargement window.

–**bg** *colororpixelvalue*
: This option specifies the name of the color to be used as the background of the enlargement window. If the name begins with a percent size (%), it is interpreted to be an absolute pixel value. This is useful when displaying large areas since pixels that are the same color as the background do not need to be painted in the enlargement. The default is to use the BlackPixel of the screen.

–**fn** *fontname*
: This option specifies the name of a font to use when displaying pixel values (used when Button1 is pressed in the enlargement window).

–**z** This option indicates that the server should be grabbed during the dynamics and the call to XGetImage. This is useful for ensuring that clients don't change their state as a result of entering or leaving them with the pointer.

X DEFAULTS

The **xmag_multivis** program uses the following X resources:

geometry (class **Geometry**)
: Specifies the size and / or location of the enlargement window.

source (class **Source**)
: Specifies the size and / or location of the source region on the screen.

magnification (class **Magnification**)
: Specifies the enlargement factor.

borderWidth (class **BorderWidth**)
: Specifies the border width in pixels.

borderColor (class **BorderColor**)
: Specifies the color of the border.

background (class **Background**)
: Specifies the color or pixel value to be used for the background of the enlargement window.

font (class **Font**)
: Specifies the name of the font to use when displaying pixel values when the user presses Button1 in the enlargement window.

SEE ALSO

xwd(1), **xmag**(1)

BUGS

On displays that export multiple visuals, if the selected region needs to be composited from each window, **xmag_multivis** insists on a TrueColor 24 bit window to display the selected region, and exits with a failure if such a window cannot be created. It would be nice, instead if **xmag_multivis** would determine the best possible visual to display the image, or instead tried to display the image in a visual selected by the user in a command line option.

Because the window size equals the source size times the magnification, you only need to specify two of the three parameters. This can be confusing.

Being able to drag the pointer around and see a dynamic display would be very nice.

Another possible interface would be for the user to drag out the desired area to be enlarged.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.

MultiVisual code: Copyright (c) 1990-92 by Sun Microsystems, Inc.

AUTHOR

Jim Fulton, MIT X Consortium

MultiVisual code: Milind Pansare, Sun Microsystems, Inc.

NAME

xmakemap – make a keyboard mapping to be used as input to xmodmap

SYNOPSIS

xmakemap

DESCRIPTION

xmakemap will produce a keyboard mapping in a form that is suitable as input to the **xmodmap**(1) command. **xmakemap** writes its output to stdout. Typical usage of **xmakemap** is to redirect output from **xmakemap** to a file (e.g., $HOME/.xmodmaprc), edit this file to make necessary customizations, and then run **xmodmap** on the file (i.e., $HOME/.xmodmaprc).

For example:

```
      xmakemap > $HOME/.xmodmaprc
      [... make necessary customization to $HOME/.xmodmaprc ...]
      xmodmap $HOME/.xmodmaprc
```

If you would like your keytable to be customized each time OpenWindows is run, you should placed the appropriate command in $HOME/.xinitrc (copy /usr/openwin/lib/Xinitrc to $HOME/.xinitrc if it does not already exist).

For example:

```
      if [ -f $HOME/.xmodmaprc ]; then
          xmodmap $HOME/.xmodmaprc
      fi
```

NOTES

OpenWindows (i.e., an X server) must be running to use **xmakemap**.
Read "Notes" in $HOME/.xmodmaprc.

SEE ALSO

xmodmap(1)

NAME

xman – manual page display program for the X Window System

SYNOPSIS

xman [–*options* . . .]

DESCRIPTION

xman is a manual page browser. The default size of the initial **xman** window is small so that you can leave it running throughout your entire login session. In the initial window there are three options: *Help* will pop up a window with on-line help, *Quit* will exit, and *Manual Page* will pop up a window with a manual page browser in it. Typing Control-S will pop up a window prompting for a specific manual page to display. You may display more than one manual page browser window at a time from a single execution of **xman**.

For further information on using **xman**, please read the on-line help information. Most of this manual will discuss customization of **xman**.

OPTIONS

Xman supports all standard Toolkit command line arguments (see **X11**(7)). The following additional arguments are supported.

–**helpfile** *filename*
: Specifies a helpfile to use other than the default.

–**bothshown**
: Allows both the manual page and manual directory to be on the screen at the same time.

–**notopbox**
: Starts without the Top Menu with the three buttons in it.

–**geometry** *WxH+X+Y*
: Sets the size and location of the Top Menu with the three buttons in it.

–**pagesize** *WxH+X+Y*
: Sets the size and location of all the Manual Pages.

CUSTOMIZING XMAN

Xman allows customization of both the directories to be searched for manual pages, and the name that each directory will map to in the *Sections* menu. **Xman** determines which directories it will search by reading the *MANPATH* environment variable. If no *MANPATH* is found then the directory is /usr/man is searched on POSIX systems. This environment is expected to be a colon-separated list of directories for **xman** to search.

```
setenv MANPATH /mit/kit/man:/usr/man
```

By default, **xman** will search each of the following directories (in each of the directories specified in the users MANPATH) for manual pages. If manual pages exist in that directory then they are added to list of manual pages for the corresponding menu item. A menu item is only displayed for those sections that actually contain manual pages.

Directory	Section Name
---------	------------
man1	(1) User Commands
man2	(2) System Calls

man3	(3) Subroutines
man4	(4) Devices
man5	(5) File Formats
man6	(6) Games
man7	(7) Miscellaneous
man8	(8) Sys. Administration
manl	(l) Local
mann	(n) New
mano	(o) Old

For instance, a user has three directories in her manual path and each contain a directory called *man3*. All these manual pages will appear alphabetically sorted when the user selects the menu item called *(3) Subroutines*. If there is no directory called *mano* in any of the directories in her MANPATH, or there are no manual pages in any of the directories called *mano* then no menu item will be displayed for the section called *(o) Old*.

THE MANDESC FILE

By using the *mandesc* file a user or system manager is able to more closely control which manual pages will appear in each of the sections represented by menu items in the *Sections* menu. This functionality is only available on a section by section basis, and individual manual pages may not be handled in this manner. (Although generous use of symbolic links — see **ln**(1) — will allow almost any configuration you can imagine.)

The format of the mandesc file is a character followed by a label. The character determines which of the sections will be added under this label. For instance suppose that you would like to create an extra menu item that contains all programmer subroutines. This label should contain all manual pages in both sections two and three. The *mandesc* file would look like this:

```
2Programmer Subroutines
3Programmer Subroutines
```

This will add a menu item to the *Sections* menu that would bring up a listing of all manual pages in sections two and three of the Programmers Manual. Since the label names are *exactly* the same they will be added to the same section. Note, however, that the original sections still exist.

If you want to completely ignore the default sections in a manual directory then add the line:

```
no default sections
```

anywhere in your mandesc file. This keeps **xman** from searching the default manual sections *In that directory only*. As an example, suppose you want to do the same thing as above, but you don’t think that it is useful to have the *System Calls* or *Subroutines* sections any longer. You would need to duplicate the default entries, as well as adding your new one.

no default sections
1(1) User Commands
2Programmer Subroutines
3Programmer Subroutines
4(4) Devices
5(5) File Formats
6(6) Games
7(7) Miscellaneous
8(8) Sys. Administration
l(l) Local
n(n) New
o(o) Old

Xman will read any section that is of the from *man<character>*, where <character> is an upper or lower case letter (they are treated distinctly) or a numeral (0-9). Be warned, however, that **man**(1) and **catman**(1M) will not search directories that are non-standard.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **xman**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xman xman       (This widget is never used)
        TopLevelShell  topbox
                Form  form
                        Label  topLabel
                        Command  helpButton
                        Command  quitButton
                        Command  manpageButton
                TransientShell  search
                        DialogWidgetClass  dialog
                                Label  label
                                Text  value
                                Command  manualPage
                                Command  apropos
                                Command  cancel
                TransientShell  pleaseStandBy
                        Label  label
        TopLevelShell  manualBrowser
                Paned  Manpage_Vpane
                        Paned  horizPane
                                MenuButton  options
                                MenuButton  sections
                                Label  manualBrowser
                        Viewport  directory
                                List  directory
```

```
                    List directory
                    .
                    . (one for each section,
                    .  created on the fly)
                    .
            ScrollByLine manualPage
        SimpleMenu optionMenu
            SmeBSB displayDirectory
            SmeBSB displayManualPage
            SmeBSB help
            SmeBSB search
            SmeBSB showBothScreens
            SmeBSB removeThisManpage
            SmeBSB openNewManpage
            SmeBSB showVersion
            SmeBSB quit
        SimpleMenu sectionMenu
            SmeBSB <name of section>
                    .
                    . (one for each section)
                    .
        TransientShell search
            DialogWidgetClass dialog
                Label label
                Text value
                Command manualPage
                Command apropos
                Command cancel
        TransientShell pleaseStandBy
            Label label
        TransientShell likeToSave
            Dialog dialog
                Label label
                Text value
                Command yes
                Command no
    TopLevelShell help
        Paned Manpage_Vpane
            Paned horizPane
                MenuButton options
                MenuButton sections
                Label manualBrowser
            ScrollByLine manualPage
        SimpleMenu optionMenu
            SmeBSB displayDirectory
```

```
SmeBSB  displayManualPage
SmeBSB  help
SmeBSB  search
SmeBSB  showBothScreens
SmeBSB  removeThisManpage
SmeBSB  openNewManpage
SmeBSB  showVersion
SmeBSB  quit
```

APPLICATION RESOURCES

xman has the following application-specific resources which allow customizations unique to **xman**.

manualFontNormal (Class **Font**)
The font to use for normal text in the manual pages.

manualFontBold (Class **Font**)
The font to use for bold text in the manual pages.

manualFontItalic (Class **Font**)
The font to use for italic text in the manual pages.

directoryFontNormal (Class **Font**)
The font to use for the directory text.

bothShown (Class **Boolean**)
Either ‘true’ or ‘false,’ specifies whether or not you want both the directory and the manual page shown at start up.

directoryHeight (Class **DirectoryHeight**)
The height in pixels of the directory, when the directory and the manual page are shown simultaneously.

topCursor (Class **Cursor**)
The cursor to use in the top box.

helpCursor (Class **Cursor**)
The cursor to use in the help window.

manpageCursor (Class **Cursor**)
The cursor to use in the manual page window.

searchEntryCursor (Class **Cursor**)
The cursor to use in the search entry text widget.

pointerColor (Class **Foreground**)
This is the color of all the cursors (pointers) specified above. The name was chosen to be compatible with xterm.

helpFile (Class **File**)
Use this rather than the system default helpfile.

topBox (Class **Boolean**)
Either ‘true’ or ‘false,’ determines whether the top box (containing the help, quit and manual page buttons) or a manual page is put on

the screen at start-up. The default is true.

verticalList (Class **Boolean**)
Either ‘true’ or ‘false,’ determines whether the directory listing is vertically or horizontally organized. The default is horizontal (false).

GLOBAL ACTIONS

Xman defines all user interaction through global actions. This allows the user to modify the translation table of any widget, and bind any event to the new user action. The list of actions supported by **xman** are:

GotoPage(*page***)** When used in a manual page display window this will allow the user to move between a directory and manual page display. The *page* argument can be either **Directory** or **ManualPage**.

Quit() This action may be used anywhere, and will exit **xman**.

Search(*type***,** *action***)** Only useful when used in a search popup, this action will cause the search widget to perform the named search type on the string in the search popup’s value widget. This action will also pop down the search widget. The *type* argument can be either **Apropos**, **Manpage** or **Cancel**. If an *action* of **Open** is specified then **xman** will open a new manual page to display the results of the search, otherwise **xman** will attempt to display the results in the parent of the search popup.

PopupHelp() This action may be used anywhere, and will popup the help widget.

PopupSearch() This action may be used anywhere except in a help window. It will cause the search popup to become active and visible on the screen, allowing the user search for a manual page.

CreateNewManpage() This action may be used anywhere, and will create a new manual page display window.

RemoveThisManpage()
This action may be used in any manual page or help display window. When called it will remove the window, and clean up all resources associated with it.

SaveFormattedPage(*action***)**
This action can only be used in the **likeToSave** popup widget, and tells **xma**n whether to **Save** or **Cancel** a save of the manual page that has just been formatted.

ShowVersion() This action may be called from any manual page or help display window, and will cause the informational display line to show the current version of **xman**.

FILES

<manpath directory> / man*<character>*

<manpath directory> / cat*<character>*

<manpath directory> / mandesc

/usr/openwin/lib/app-defaults/Xman	specifies required resources
/tmp	*Xman* creates temporary files in /tmp for all unformatted man pages and all apropos searches.

SEE ALSO

X11(7) **man**(1), **apropos**(1), **catman**(1M), *Athena Widget Set*

ENVIRONMENT

DISPLAY	the default host and display to use.
MANPATH	the search path for manual pages. Directories are separated by colons (e.g. /usr/man:/mit/kit/man:/foo/bar/man).
XENVIRONMENT	to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.
XAPPLRESDIR	A string that will have ''Xman'' appended to it. This string will be the full path name of a user app-defaults file to be merged into the resource database after the system app-defaults file, and before the resources that are attached to the display.

COPYRIGHT

Copyright 1988 by Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHORS

Chris Peterson, MIT X Consortium from the V10 version written by Barry Shein formerly of Boston University.

NAME

xmh – send and read mail with an X interface to MH

SYNOPSIS

xmh [–path *mailpath*] [–initial *foldername*] [–flag] [–*toolkitoption* ...]

DESCRIPTION

The **xmh** program provides a graphical user interface to the *MH* Message Handling System. To actually do things with your mail, it makes calls to the *MH* package. Electronic mail messages may be composed, sent, received, replied to, forwarded, sorted, and stored in folders. **xmh** provides extensive mechanism for customization of the user interface.

This document introduces many aspects of the Athena Widget Set.

OPTIONS

–**path** *directory*
: This option specifies an alternate collection of mail folders in which to process mail. The directory is specified as an absolute pathname. The default mail path is the value of the Path component in the *MH* profile, which is determined by the **MH** environment variable and defaults to $HOME/.mh_profile. $HOME/Mail will be used as the path if the *MH* Path is not given in the profile.

–**initial** *folder*
: This option specifies an alternate folder which may receive new mail and is initially opened by **xmh.** The default initial folder is ‘‘inbox’’.

–**flag**
: This option will cause **xmh** to change the appearance of appropriate folder buttons and to request the window manager to change the appearance of the **xmh** icon when new mail has arrived. By default, **xmh** will change the appearance of the ‘‘inbox’’ folder button when new mail is waiting. The application-specific resource **checkNewMail** can be used to turn off this notification, and the –**flag** option will still override it.

These three options have corresponding application-specific resources, **MailPath**, **InitialFolder**, and **MailWaitingFlag**, which can be specified in a resource file.

The standard toolkit command line options are given in **X11**(7).

INSTALLATION

xmh requires that the user is already set up to use *MH*, version 6. To do so, see if there is a file called .mh_profile in your home directory. If it exists, check to see if it contains a line that starts with ‘‘Current-Folder’’. If it does, you’ve been using version 4 or earlier of *MH*; to convert to version 6, you must remove that line. (Failure to do so causes spurious output to stderr, which can hang **xmh** depending on your setup.)

If you do not already have a .mh_profile, you can create one (and everything else you need) by typing ‘‘inc’’ to the shell. You should do this before using **xmh** to incorporate new mail.

Much of the user interface of **xmh** is configured in the *Xmh* application class defaults file; if this file was not installed properly a warning message will appear when **xmh** is used. **xmh** is backwards compatible with the R4 application class defaults file.

The default value of the SendBreakWidth resource has changed since R4.

BASIC SCREEN LAYOUT

xmh starts out with a single window, divided into four major areas:

– Six buttons with pull-down command menus.
– A collection of buttons, one for each top level folder. New users of *MH* will have two folders, ‘‘drafts’’ and ‘‘inbox’’.
– A listing, or Table of Contents, of the messages in the open folder. Initially, this will show the messages in ‘‘inbox’’.
– A view of one of your messages. Initially this is blank.

XMH AND THE ATHENA WIDGET SET

xmh uses the X Toolkit Intrinsics and the Athena Widget Set. Many of the features described below (scrollbars, buttonboxes, etc.) are actually part of the Athena Widget Set, and are described here only for completeness. For more information, see the Athena Widget Set documentation.

SCROLLBARS

Some parts of the main window will have a vertical area on the left containing a grey bar. This area is a *scrollbar*. They are used whenever the data in a window takes up more space than can be displayed. The grey bar indicates what portion of your data is visible. Thus, if the entire length of the area is grey, then you are looking at all your data. If only the first half is grey, then you are looking at the top half of your data. The message viewing area will have a horizontal scrollbar if the text of the message is wider than the viewing area.

You can use the pointer in the scrollbar to change what part of the data is visible. If you click with pointer button 2, the top of the grey area will move to where the pointer is, and the corresponding portion of data will be displayed. If you hold down pointer button 2, you can drag around the grey area. This makes it easy to get to the top of the data: just press with button 2, drag off the top of the scrollbar, and release.

If you click with button 1, then the data to the right of the pointer will scroll to the top of the window. If you click with pointer button 3, then the data at the top of the window will scroll down to where the pointer is.

BUTTONBOXES, BUTTONS, AND MENUS

Any area containing many words or short phrases, each enclosed in a rectangular or rounded boundary, is called a *buttonbox*. Each rectangle or rounded area is actually a button that you can press by moving the pointer onto it and pressing pointer button 1. If a given buttonbox has more buttons in it than can fit, it will be displayed with a scrollbar, so you can always scroll to the button you want.

Some buttons have pull-down menus. Pressing the pointer button while the pointer is over one of these buttons will pull down a menu. Continuing to hold the button down while moving the pointer over the menu, called dragging the pointer, will highlight each selectable item on the menu as the pointer passes over it. To select an item in the menu, release the pointer button while the item is highlighted.

ADJUSTING THE RELATIVE SIZES OF AREAS

If you're not satisfied with the sizes of the various areas of the main window, they can easily be changed. Near the right edge of the border between each region is a black box, called a *grip*. Simply point to that grip with the pointer, press a pointer button, drag up or down, and release. Exactly what happens depends on which pointer button you press.

If you drag with the pointer button 2, then only that border will move. This mode is simplest to understand, but is the least useful.

If you drag with pointer button 1, then you are adjusting the size of the window above. **xmh** will attempt to compensate by adjusting some window below it.

If you drag with pointer button 3, then you are adjusting the size of the window below. **xmh** will attempt to compensate by adjusting some window above it.

All windows have a minimum and maximum size; you will never be allowed to move a border past the point where it would make a window have an invalid size.

PROCESSING YOUR MAIL

This section will define the concepts of the selected folder, current folder, selected message(s), current message, selected sequence, and current sequence. Each **xmh** command is introduced.

For use in customization, action procedures corresponding to each command are given; these action procedures can be used to customize the user interface, particularly the keyboard accelerators and the functionality of the buttons in the optional button box created by the application resource **CommandButtonCount**.

FOLDERS AND SEQUENCES

A folder contains a collection of mail messages, or is empty. **xmh** supports folders with one level of subfolders.

The selected folder is whichever foldername appears in the bar above the folder buttons. Note that this is not necessarily the same folder that is currently being viewed. To change the selected folder, just press on the desired folder button with pointer button 1; if that folder has subfolders, select a folder from the pull-down menu.

The Table of Contents, or toc, lists the messages in the viewed folder. The title bar above the Table of Contents displays the name of the viewed folder.

The toc title bar also displays the name of the viewed sequence of messages within the viewed folder. Every folder has an implicit ‘‘all’’ sequence, which contains all the messages in the folder, and initially the toc title bar will show ‘‘inbox:all’’.

FOLDER COMMANDS

The *Folder* command menu contains commands of a global nature:

Open Folder

Display the data in the selected folder. Thus, the selected folder also becomes the viewed folder. The action procedure corresponding to this command is **XmhOpenFolder(**[*foldername*]**)**. It takes an optional argument as the name of a folder to select and open; if no folder is specified, the selected folder is opened. It may be specified as part of an event translation from a folder menu button or from a folder menu, or as a binding of a keyboard accelerator to any widget other than the folder menu buttons or the folder menus.

Open Folder in New Window
Displays the selected folder in an additional main window. Note, however, that you cannot reliably display the same folder in more than one window at a time, although **xmh** will not prevent you from trying. The corresponding action is **XmhOpenFolderInNewWindow()**.

Create Folder
Create a new folder. You will be prompted for a name for the new folder; to enter the name, move the pointer to the blank box provided and type. Subfolders are created by specifying the parent folder, a slash, and the subfolder name. For example, to create a folder named ''xmh'' which is a subfolder of an existing folder named ''clients'', type ''clients/xmh''. Click on the Okay button when finished, or just type Return; click on Cancel to cancel this operation. The action corresponding to Create Folder is **XmhCreateFolder()**.

Delete Folder
Destroy the selected folder. You will be asked to confirm this action (see CONFIRMATION WINDOWS). Destroying a folder will also destroy any subfolders of that folder. The corresponding action is **XmhDeleteFolder()**.

Close Window
Exits **xmh**, after first confirming that you won't lose any changes; or, if selected from any additional **xmh** window, simply closes that window. The corresponding action is **XmhClose()**.

HIGHLIGHTED MESSAGES, SELECTED MESSAGES, CURRENT MESSAGE

It is possible to highlight a set of adjacent messages in the area of the Table of Contents. To highlight a message, click on it with pointer button 1. To highlight a range of messages, click on the first one with pointer button 1 and on the last one with pointer button 3; or press pointer button 1, drag, and release. To extend a range of selected messages, use pointer button 3. To highlight all messages in the table of contents, click rapidly three times with pointer button 1. To cancel any selection in the table of contents, click rapidly twice.

The selected messages are the same as the highlighted messages, if any. If no messages are highlighted, then the selected messages are considered the same as the current message.

The current message is indicated by a '+' next to the message number. It usually corresponds to the message currently being viewed. Upon opening a new folder, for example, the current message will be different from the viewed message. When a message is viewed, the title bar above the view will identify the message.

TABLE OF CONTENTS COMMANDS

The *Table of Contents* command menu contains commands which operate on the open, or viewed, folder.

Incorporate New Mail
Add any new mail received to viewed folder, and set the current message to be the first new message. This command is selectable in the menu and will execute only if the viewed folder is allowed to receive new mail. By default, only ‘‘inbox’’ is allowed to incorporate new mail. The corresponding action is **XmhIncorporateNewMail()**.

Commit Changes Execute all deletions, moves, and copies that have been marked in this folder. The corresponding action is **XmhCommitChanges()**.

Pack Folder Renumber the messages in this folder so they start with 1 and increment by 1. The corresponding action is **XmhPackFolder()**.

Sort Folder Sort the messages in this folder in chronological order. (As a side effect, this may also pack the folder.) The corresponding action is **XmhSortFolder()**.

Rescan Folder Rebuild the list of messages. This can be used whenever you suspect that **xmh**’s idea of what messages you have is wrong. (In particular, this is necessary if you change things using straight *MH* commands without using **xmh**.) The corresponding action is **XmhForceRescan()**.

MESSAGE COMMANDS

The *Message* command menu contains commands which operate on the selected message(s), or if there are no selected messages, the current message.

Compose Message Composes a new message. A new window will be brought up for composition; a description of it is given in the COMPOSITION WINDOWS section below. This command does not affect the current message. The corresponding action is **XmhComposeMessage()**.

View Next Message
View the first selected message. If no messages are highlighted, view the current message. If current message is already being viewed, view the first unmarked message after the current message. The corresponding action is **XmhViewNextMessage()**.

View Previous View the last selected message. If no messages are highlighted, view the current message. If current message is already being viewed, view the first unmarked message before the current message. The corresponding action is **XmhViewPrevious()**.

Delete
Mark the selected messages for deletion. If no messages are highlighted, mark the current message for deletion and automatically display the next unmarked message. The corresponding action is **XmhMarkDeleted()**.

Move
Mark the selected messages to be moved into the currently selected folder. (If the selected folder is the same as the viewed folder, this command will just beep.) If no messages are highlighted, mark the current message to be moved and display the next unmarked message. The corresponding action is **XmhMarkMove()**.

Copy as Link
Mark the selected messages to be copied into the selected folder. (If the selected folder is the same as the viewed folder, this command will just beep.) If no messages are highlighted, mark the current message to be copied. Note that messages are actually linked, not copied; editing a message copied by **xmh** will affect all copies of the message. The corresponding action is **XmhMarkCopy()**.

Unmark
Remove any of the above three marks from the selected messages, or the current message, if none are highlighted. The corresponding action is **XmhUnmark()**.

View in New
Create a new window containing only a view of the first selected message, or the current message, if none are highlighted. The corresponding action is **XmhViewInNewWindow()**.

Reply
Create a composition window in reply to the first selected message, or the current message, if none are highlighted. The corresponding action is **XmhReply()**.

Forward
Create a composition window whose body is initialized to contain an encapsulation of of the selected messages, or the current message if none are highlighted. The corresponding action is **XmhForward()**.

Use as Composition
Create a composition window whose body is initialized to be the contents of the first selected message, or the current message if none are selected. Any changes you make in the composition will be saved in a new message in the ‘‘drafts’’ folder, and will not change the original message. However, there is an exception to this rule. If the message to be used as composition was selected from the ‘‘drafts’’ folder, (see BUGS), the changes will be reflected in the original message (see COMPOSITION WINDOWS). The action procedure corresponding to this command is **XmhUseAsComposition()**.

Print
Print the selected messages, or the current message if none are selected. **xmh** normally prints by invoking the **enscript** command, but this can be customized with the **xmh** application-specific resource **PrintCommand**. The corresponding action is **XmhPrint()**.

SEQUENCE COMMANDS

The *Sequence* command menu contains commands pertaining to message sequences (See MESSAGE-SEQUENCES), and a list of the message-sequences defined for the currently viewed folder. The selected message-sequence is indicated by a check mark in its entry in the margin of the menu. To change the selected message-sequence, select a new message-sequence from the sequence menu.

Pick Messages Define a new message-sequence. The corresponding action is **XmhPickMessages()**.

The following menu entries will be sensitive only if the current folder has any message-sequences other than the ''all'' message-sequence.

Open Sequence Change the viewed sequence to be the same as the selected sequence. The corresponding action is **XmhOpenSequence()**.

Add to Sequence Add the selected messages to the selected sequence. The corresponding action is **XmhAddToSequence()**.

Remove from Sequence
Remove the selected messages from the selected sequence. The corresponding action is **XmhRemoveFromSequence()**.

Delete Sequence Remove the selected sequence entirely. The messages themselves are not affected; they simply are no longer grouped together to define a message-sequence. The corresponding action is **XmhDeleteSequence()**.

VIEW COMMANDS

Commands in the *View* menu and in the buttonboxes of view windows (which result from the *Message* menu command **View In New**) correspond in functionality to commands of the same name in the *Message* menu, but they operate on the viewed message rather than the selected messages or current message.

Close Window When the viewed message is in a separate view window, this command will close the view, after confirming the status of any unsaved edits. The corresponding action procedure is **XmhCloseView()**.

Reply Create a composition window in reply to the viewed message. The related action procedure is **XmhViewReply()**.

Forward Create a composition window whose body is initialized contain an encapsulation of the viewed message. The corresponding action is **XmhViewForward()**.

Use As Composition
Create a composition window whose body is initialized to be the contents of the viewed message. Any changes made in the composition window will be saved in a new message in the ''drafts'' folder, and will not change the original message. An exception: if the viewed message was selected from the ''drafts'' folder, (see BUGS)

the original message is edited. The action procedure corresponding to this command is **XmhViewUseAsComposition()**.

Edit Message This command enables the direct editing of the viewed message. The action procedure is **XmhEditView()**.

Save Message This command is insensitive until the message has been edited; when activated, edits will be saved to the original message in the view. The corresponding action is **XmhSaveView()**.

Print Print the viewed message. **xmh** prints by invoking the **enscript** command, but this can be customized with the application-specific resource **PrintCommand**. The corresponding action procedure is **XmhPrintView()**.

Delete Marks the viewed message for deletion. The corresponding action procedure is **XmhViewMarkDelete()**.

OPTIONS

The *Options* menu contains one entry.

Read in Reverse

When selected, a check mark appears in the margin of this menu entry. Read in Reverse will switch the meaning of the next and previous messages, and will increment to the current message marker in the opposite direction. This is useful if you want to read your messages in the order of most recent first. The option acts as a toggle; select it from the menu a second time to undo the effect. The check mark appears when the option is selected.

COMPOSITION WINDOWS

Composition windows are created by selecting **Compose Message** from the *Message* command menu, or by selecting **Reply** or **Forward** or **Use as Composition** from the *Message* or *View* command menu. These are used to compose mail messages. Aside from the normal text editing functions, there are six command buttons associated with composition windows:

Close Window Close this composition window. If changes have been made since the most recent Save or Send, you will be asked to confirm losing them. The corresponding action is **XmhCloseView()**.

Send Send this composition. The corresponding action is **XmhSend()**.

New Headers Replace the current composition with an empty message. If changes have been made since the most recent Send or Save, you will be asked to confirm losing them. The corresponding action is **XmhResetCompose()**.

Compose Message Bring up another new composition window. The corresponding action is **XmhComposeMessage()**.

Save Message Save this composition in your drafts folder. Then you can safely close the composition. At some future date, you can continue working on the composition by opening the drafts folder, selecting the message, and using the ''Use as Composition'' command. The corresponding action is **XmhSave()**.

Insert Insert a related message into the composition. If the composition window was created with a ''Reply'' command, the related message is the message being replied to, otherwise no related message is defined and this button is insensitive. The message may be filtered before being inserted; see **ReplyInsertFilter** under APPLICATION RESOURCES for more information. The corresponding action is **XmhInsert()**.

ACCELERATORS

Accelerators are shortcuts. They allow you to invoke commands without using the menus, either from the keyboard or by using the pointer.

xmh defines pointer accelerators for common actions: To select and view a message with a single click, use pointer button 2 on the message's entry in the table of contents. To select and open a folder or a sequence in a single action, make the folder or sequence selection with pointer button 2.

To mark the highlighted messages, or current message if none have been highlighted, to be moved to a folder in a single action, use pointer button 3 to select the target folder and simultaneously mark the messages. Similarly, selecting a sequence with pointer button 3 will add the highlighted or current message(s) to that sequence. In both of these operations, the selected folder or sequence and the viewed folder or sequence are not changed.

xmh defines the following keyboard accelerators over the surface of the main window, except in the view area while editing a message:

Meta-I	Incorporate New Mail
Meta-C	Commit Changes
Meta-R	Rescan Folder
Meta-P	Pack Folder
Meta-S	Sort Folder
Meta-space	View Next Message
Meta-c	Mark Copy
Meta-d	Mark Deleted
Meta-f	Forward the selected or current message
Meta-m	Mark Move
Meta-n	View Next Message
Meta-p	View Previous Message
Meta-r	Reply to the selected or current message
Meta-u	Unmark

Ctrl-V	Scroll the table of contents forward
Meta-V	Scroll the table of contents backward
Ctrl-v	Scroll the view forward
Meta-v	Scroll the view backward

TEXT EDITING COMMANDS

All of the text editing commands are actually defined by the Text widget in the Athena Widget Set. The commands may be bound to different keys than the defaults described below through the X Toolkit Intrinsics key re-binding mechanisms. See the X Toolkit Intrinsics and the Athena Widget Set documentation for more details.

Whenever you are asked to enter any text, you will be using a standard text editing interface. Various control and meta keystroke combinations are bound to a somewhat Emacs-like set of commands. In addition, the pointer buttons may be used to select a portion of text or to move the insertion point in the text. Pressing pointer button 1 causes the insertion point to move to the pointer. Double-clicking button 1 selects a word, triple-clicking selects a line, quadruple-clicking selects a paragraph, and clicking rapidly five times selects everything. Any selection may be extended in either direction by using pointer button 3.

In the following, a *line* refers to one displayed row of characters in the window. A *paragraph* refers to the text between carriage returns. Text within a paragraph is broken into lines for display based on the current width of the window. When a message is sent, text is broken into lines based upon the values of the **SendBreakWidth** and **SendWidth** application-specific resources.

The following keystroke combinations are defined:

Ctrl-a	Beginning Of Line	Meta-b	Backward Word
Ctrl-b	Backward Character	Meta-f	Forward Word
Ctrl-d	Delete Next Character	Meta-i	Insert File
Ctrl-e	End Of Line	Meta-k	Kill To End Of Paragraph
Ctrl-f	Forward Character	Meta-q	Form Paragraph
Ctrl-g	Multiply Reset	Meta-v	Previous Page
Ctrl-h	Delete Previous Character	Meta-y	Insert Current Selection
Ctrl-j	Newline And Indent	Meta-z	Scroll One Line Down
Ctrl-k	Kill To End Of Line	Meta-d	Delete Next Word
Ctrl-l	Redraw Display	Meta-D	Kill Word
Ctrl-m	Newline	Meta-h	Delete Previous Word
Ctrl-n	Next Line	Meta-H	Backward Kill Word
Ctrl-o	Newline And Backup	Meta-<	Beginning Of File
Ctrl-p	Previous Line	Meta->	End Of File
Ctrl-r	Search / Replace Backward	Meta-]	Forward Paragraph
Ctrl-s	Search / Replace Forward	Meta-[	Backward Paragraph
Ctrl-t	Transpose Characters		

Ctrl-u	Multiply by 4	Meta-Delete	Delete Previous Word
Ctrl-v	Next Page	Meta-Shift Delete	Kill Previous Word
Ctrl-w	Kill Selection	Meta-Backspace	Delete Previous Word
Ctrl-y	Unkill	Meta-Shift Backspace	Kill Previous Word
Ctrl-z	Scroll One Line Up		

In addition, the pointer may be used to copy and paste text:

Button 1 Down	Start Selection
Button 1 Motion	Adjust Selection
Button 1 Up	End Selection (copy)
Button 2 Down	Insert Current Selection (paste)
Button 3 Down	Extend Current Selection
Button 3 Motion	Adjust Selection
Button 3 Up	End Selection (copy)

CONFIRMATION DIALOG BOXES

Whenever you press a button that may cause you to lose some work or is otherwise dangerous, a popup dialog box will appear asking you to confirm the action. This window will contain an ‘‘Abort’’ or ‘‘No’’ button and a ‘‘Confirm’’ or ‘‘Yes’’ button. Pressing the ‘‘No’’ button cancels the operation, and pressing the ‘‘Yes’’ will proceed with the operation.

Some dialog boxes contain messages from *MH*. Occasionally when the message is more than one line long, not all of the text will be visible. Clicking on the message field will cause the dialog box to resize so that you can read the entire message.

MESSAGE-SEQUENCES

An *MH* message sequence is just a set of messages associated with some name. They are local to a particular folder; two different folders can have sequences with the same name. The sequence named ‘‘all’’ is predefined in every folder; it consists of the set of all messages in that folder. As many as nine sequences may be defined for each folder, including the predefined ‘‘all’’ sequence. (The sequence ‘‘cur’’ is also usually defined for every folder; it consists of only the current message. **xmh** hides ‘‘cur’’ from the user, instead placing a ‘‘+’’ by the current message. Also, **xmh** does not support *MH*’s‘‘unseen’’ sequence, so that one is also hidden from the user.)

The message sequences for a folder (including one for ‘‘all’’) are displayed in the ‘‘Sequence’’ menu, below the sequence commands. The table of contents (also known as the ‘‘toc’’) is at any one time displaying one message sequence. This is called the ‘‘viewed sequence’’, and its name will be displayed in the toc title bar after the folder name. Also, at any time one of the sequences in the menu will have a check mark next to it. This is called the ‘‘selected sequence’’. Note that the viewed sequence and the selected sequence are not necessarily the same. (This all pretty much corresponds to the way folders work.)

The **Open Sequence**, **Add to Sequence**, **Remove from Sequence**, and **Delete Sequence** commands are active only if the viewed folder contains message-sequences other than ‘‘all’’ sequence.

Note that none of the above actually affect whether a message is in the folder. Remember that a sequence is a set of messages within the folder; the above operations just affect what messages are in that set.

To create a new sequence, select the ‘‘Pick’’ menu entry. A new window will appear, with lots of places to enter text. Basically, you can describe the sequence’s initial set of messages based on characteristics of the message. Thus, you can define a sequence to be all the messages that were from a particular person, or with a particular subject, and so on. You can also connect things up with boolean operators, so you can select all things from ‘‘weissman’’ with a subject containing ‘‘xmh’’.

The layout should be fairly obvious. The simplest cases are the easiest: just point to the proper field and type. If you enter in more than one field, it will only select messages which match all non-empty fields.

The more complicated cases arise when you want things that match one field or another one, but not necessarily both. That’s what all the ‘‘or’’ buttons are for. If you want all things with subjects that include ‘‘xmh’’ or ‘‘xterm’’, just press the ‘‘or’’ button next to the ‘‘Subject:’’ field. Another box will appear where you can enter another subject.

If you want all things either from ‘‘weissman’’ or with subject ‘‘xmh’’, but not necessarily both, select the ‘‘–Or–’’ button. This will essentially double the size of the form. You can then enter ‘‘weissman’’ in a from: box on the top half, and ‘‘xmh’’ in a subject: box on the lower part.

If you select the ‘‘Skip’’ button, then only those messages that *don’t* match the fields on that row are included.

Finally, in the bottom part of the window will appear several more boxes. One is the name of the sequence you’re defining. (It defaults to the name of the selected sequence when ‘‘Pick’’ was pressed, or to ‘‘temp’’ if ‘‘all’’ was the selected sequence.) Another box defines which sequence to look through for potential members of this sequence; it defaults to the viewed sequence when ‘‘Pick’’ was pressed.

Two more boxes define a date range; only messages within that date range will be considered. These dates must be entered in RFC 822-style format: each date is of the form ‘‘dd mmm yy hh:mm:ss zzz’’, where dd is a one or two digit day of the month, mmm is the three-letter abbreviation for a month, and yy is a year. The remaining fields are optional: hh, mm, and ss specify a time of day, and zzz selects a time zone. Note that if the time is left out, it defaults to midnight; thus if you select a range of ‘‘7 nov 86’’ – ‘‘8 nov 86’’, you will only get messages from the 7th, as all messages on the 8th will have arrived after midnight.

‘‘Date field’’ specifies which field in the header to look at for this date range; it defaults to ‘‘Date’’. If the sequence you’re defining already exists, you can optionally merge the old set with the new; that’s what the ‘‘Yes’’ and ‘‘No’’ buttons are all about. Finally, you can ‘‘OK’’ the whole thing, or ‘‘Cancel’’ it.

In general, most people will rarely use these features. However, it’s nice to occasionally use ‘‘Pick’’ to find some messages, look through them, and then hit ‘‘Delete Sequence’’ to put things back in their original state.

WIDGET HIERARCHY

In order to specify resources, it is useful to know the hierarchy of widgets which compose **xmh**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name. The application class name is Xmh.

The hierarchy of the main toc and view window is identical for additional toc and view windows, except that a TopLevelShell widget is inserted in the hierarchy between the application shell and the Paned widget.

```
Xmh xmh
        Paned xmh
                SimpleMenu  folderMenu
                        SmeBSB  open
                        SmeBSB  openInNew
                        SmeBSB  create
                        SmeBSB  delete
                        SmeLine  line
                        SmeBSB  close
                SimpleMenu  tocMenu
                        SmeBSB  inc
                        SmeBSB  commit
                        SmeBSB  pack
                        SmeBSB  sort
                        SmeBSB  rescan
                SimpleMenu  messageMenu
                        SmeBSB  compose
                        SmeBSB  next
                        SmeBSB  prev
                        SmeBSB  delete
                        SmeBSB  move
                        SmeBSB  copy
                        SmeBSB  unmark
                        SmeBSB  viewNew
                        SmeBSB  reply
                        SmeBSB  forward
                        SmeBSB  useAsComp
                        SmeBSB  print
                SimpleMenu  sequenceMenu
                        SmeBSB  pick
                        SmeBSB  openSeq
                        SmeBSB  addToSeq
                        SmeBSB  removeFromSeq
                        SmeBSB  deleteSeq
                        SmeLine  line
                        SmeBSB  all
                SimpleMenu  viewMenu
                        SmeBSB  reply
                        SmeBSB  forward
                        SmeBSB  useAsComp
                        SmeBSB  edit
                        SmeBSB  save
                        SmeBSB  print
                SimpleMenu  optionMenu
                        SmeBSB  reverse
                Viewport.Core  menuBox.clip
                        Box  menuBox
                                MenuButton  folderButton
                                MenuButton  tocButton
                                MenuButton  messageButton
                                MenuButton  sequenceButton
                                MenuButton  viewButton
                                MenuButton  optionButton
                Grip  grip
                Label folderTitlebar
                Grip  grip
                Viewport.Core  folders.clip
                        Box  folders
                                MenuButton  inbox
                                MenuButton  drafts
                                        SimpleMenu  menu
                                                SmeBSB <folder_name>
                                                        .
                                                        .
                                                        .

                Grip  grip
                Label  tocTitlebar
                Grip  grip
                Text toc
                        Scrollbar  vScrollbar
                Grip  grip
```

```
                Label viewTitlebar
                Grip grip
                Text view
                        Scrollbar vScrollbar
                        Scrollbar hScrollbar
```

The hierarchy of the Create Folder popup dialog box:

```
        TransientShell prompt
                Dialog dialog
                        Label label
                        Text value
                        Command okay
                        Command cancel
```

The hierarchy of the Notice dialog box, which reports messages from MH:

```
        TransientShell notice
                Dialog dialog
                        Label label
                        Text value
                        Command confirm
```

The hierarchy of the Confirmation dialog box:

```
        TransientShell confirm
                Dialog dialog
                        Label label
                        Command yes
                        Command no
```

The hierarchy of the dialog box which reports errors:

```
        TransientShell error
                Dialog dialog
                        Label label
                        Command OK
```

The hierarchy of the composition window:

```
        TopLevelShell xmh
                Paned xmh
                        Label composeTitlebar
                        Text comp
                        Viewport.Core compButtons.clip
                Box compButtons
                        Command close
                        Command send
                        Command reset
                        Command compose
                        Command save
                        Command insert
```

The hierarchy of the view window:

```
TopLevelShell xmh
        Paned xmh
                Label viewTitlebar
                Text view
                Viewport.Core viewButtons.clip
                        Box viewButtons
                                Command close
                                Command reply
                                Command forward
                                Command useAsComp
                                Command edit
                                Command save
                                Command print
                                Command delete
```

The hierarchy of the pick window:
(Unnamed widgets have no name.)

```
TopLevelShell xmh
        Paned xmh
                Label pickTitlebar
                Viewport.Core pick.clip
                        Form form
                                Form groupform
```
The first 6 rows of the pick window have identical structure:
```
                                        Form rowform
                                                Toggle
                                                Toggle
                                                Label
                                                Text
                                                Command

                                        Form rowform
                                                Toggle
                                                Toggle
                                                Text
                                                Text
                                                Command
                                        Form  rowform
                                                Command
                Viewport.core  pick.clip
                        Form  form
                                From  groupform
                                        Form  rowform
                                                Label
                                                Text
                                                Label
                                                Text
                                        Form  rowform
                                                Label
                                                Text
                                                Label
                                                Text
                                                Label
                                                Text
                                        Form  rowform
                                                Label
                                                Toggle
                                                Toggle
                                        Form  rowform
                                                Command
                                                Command
```

APPLICATION-SPECIFIC RESOURCES

The application class name is **Xmh**. Application-specific resources are listed below by name. Application-specific resource class names always begin with an upper case character, but unless noted, are otherwise identical to the instance names given below.

Any of these options may also be specified on the command line by using the X Toolkit Intrinsics resource specification mechanism. Thus, to run **xmh** showing all message headers,
% xmh –xrm '∗HideBoringHeaders:off'

If **TocGeometry**, **ViewGeometry**, **CompGeometry**, or **PickGeometry** are not specified, then the value of **Geometry** is used instead. If the resulting height is not specified (e.g., "", "=500", "+0-0"), then the default height of windows is calculated from fonts and line counts. If the width is not specified (e.g., "", "=x300", "-0+0"), then half of the display width is used. If unspecified, the height of a pick window defaults to half the height of the display.

The following resources are defined:

banner A short string that is the default label of the folder, Table of Contents, and view. The default is "xmh MIT X Consortium R5".

blockEventsOnBusy
Whether to disallow user input and show a busy cursor while **xmh** is busy processing a command. Default is true.

busyCursor
The name of the symbol used to represent the position of the pointer, displayed if **blockEventsOnBusy** is true, when **xmh** is processing a time-consuming command. The default is "watch".

busyPointerColor
The foreground color of the busy cursor. Default is XtDefaultForeground.

checkFrequency
How often to check for new mail, make checkpoints, and rescan the Table of Contents, in minutes. If **checkNewMail** is true, **xmh** checks to see if you have new mail each interval. If **makeCheckpoints** is true, checkpoints are made every fifth interval. Also every fifth interval, the Table of Contents is checked for inconsistencies with the file system, and rescanned if out of date. To prevent all of these checks from occurring, set **CheckFrequency** to 0. The default is 1. This resource is retained for backward compatibility with user resource files; see also **checkpointInterval**, **mailInterval**, and **rescanInterval**.

checkNewMail
If true, **xmh** will check at regular intervals to see if new mail has arrived for any of the top level folders and any opened subfolders. A visual indication will be given if new mail is waiting to be incorporated into a top level folder. Default is true. The interval can be adjusted with **mailInterval**.

checkpointInterval (class **Interval**)
Specifies in minutes how often to make checkpoints of volatile state, if **makeCheckpoints** is true. The default is 5 times the value of **checkFrequency**.

checkpointNameFormat
Specifies how checkpointed files are to be named. The value of this resource will be used to compose a file name by inserting the message number as a string in place of the required single occurance of ‘%d’. If the value of the resource is the empty string, or if no ‘%d’ occurs in the string, or if "%d" is the value of the resource, the default will be used instead. The default is "%d.CKP". Checkpointing is done in the folder of origin unless an absolute pathname is given. **xmh** does not assist the user in recovering checkpoints, nor does it provide for removal of the checkpoint files.

commandButtonCount
The number of command buttons to create in a button box in between the toc and the view areas of the main window. **xmh** will create these buttons with the names *button1*, *button2* and so on, in a box with the name *commandBox*. The default is 0. **xmh** users can specify labels and actions for the buttons in a private

resource file; see the section ACTIONS AND INTERFACE CUSTOMIZATION.

compGeometry
Initial geometry for windows containing compositions.

cursor The name of the symbol used to represent the pointer. Default is ‘‘left_ptr’’.

debug Whether or not to print information to stderr as **xmh** runs. Default is false.

draftsFolder
The folder used for message drafts. Default is ‘‘drafts’’.

geometry
Default geometry to use. Default is none.

hideBoringHeaders
If ‘‘on’’, then **xmh** will attempt to skip uninteresting header lines within messages by scrolling them off the top of the view. Default is ‘‘on’’.

initialFolder
Which folder to display on startup. May also be set with the command-line option –**initial**. Default is ‘‘inbox’’.

initialIncFile
The absolute path name of your incoming mail drop file. In some installations, for example those using the Post Office Protocol, no file is appropriate. In this case, **initialIncFile** should not be specified, or may be specified as the empty string, and *inc* will be invoked without a –file argument. By default, this resource has no value. This resource is ignored if **xmh** finds an *.xmhcheck* file; see the section on multiple mail drops.

mailInterval (class **Interval)**
Specifies the interval in minutes at which the mail should be checked, if **mailWaitingFlag** or **checkNewMail** is true. The default is the value of **checkFrequency**.

mailPath
The full path prefix for locating your mail folders. May also be set with the command line option, –**path**. The default is the Path component in the *MH* profile, or ‘‘$HOME / Mail’’ if none.

mailWaitingFlag
If true, **xmh** will attempt to set an indication in its icon when new mail is waiting to be retrieved. If **mailWaitingFlag** is true, then **checkNewMail** is assumed to be true as well. The –**flag** command line option is a quick way to turn on this resource.

makeCheckpoints
If true, **xmh** will attempt to save checkpoints of volatile edits. The default is false. The frequency of checkpointing is controlled by the resource **checkpointInterval**. For the location of checkpointing, see **checkpointNameFormat**.

mhPath What directory in which to find the *MH* commands. If a command isn't found in the user's path, then the path specified here is used. Default is ‘‘/usr/local/mh6’’.

newMailBitmap (class **NewMailBitmap**)
The bitmap to show in the folder button when a folder has new mail. The default is ‘‘black6’’.

newMailIconBitmap (class **NewMailBitmap**)
The bitmap suggested to the window manager for the icon when any folder has new mail. The default is ‘‘flagup’’.

noMailBitmap (class **NoMailBitmap)**
The bitmap to show in the folder button when a folder has no new mail. The default is ‘‘box6’’.

noMailIconBitmap (class **NoMailBitmap)**
The bitmap suggested to the window manager for the icon when no folders have new mail. The default is ‘‘flagdown’’.

pickGeometry
Initial geometry for pick windows.

pointerColor
The foreground color of the pointer. Default is XtDefaultForeground.

prefixWmAndIconName
Whether to prefix the window and icon name with "xmh: ". Default is true.

printCommand
An *sh* command to execute to print a message. Note that stdout and stderr must be specifically redirected. If a message or range of messages is selected for printing, the full file paths of each message file are appended to the specified print command. The default is ‘‘enscript >/dev/null 2>/dev/null’’.

replyInsertFilter
An *sh* command to be executed when the *Insert* button is activated in a composition window. The full path and filename of the source message is appended to the command before being passed to **sh**(1). The default filter is *cat*; i.e. it inserts the entire message into the composition. Interesting filters are: *sed 's/^/> /'* or *awk -e '{print "   " $0}'* or *<mh directory>/lib/mhl –form mhl.body*.

rescanInterval (class **Interval**)
How often to check the Table of Contents of currently viewed folders and of folders with messages currently being viewed, and to update the Table of Contents if **xmh** sees inconsistencies with the file system in these folders. The default is 5 times the value of **checkFrequency**.

reverseReadOrder
When true, the next message will be the message prior to the current message in the table of contents, and the previous message will be the message after the current message in the table of contents. The default is false.

sendBreakWidth
When a message is sent from **xmh**, lines longer than this value will be split into multiple lines, each of which is no longer than **SendWidth**. This value may be overridden for a single message by inserting an additional line in the message header of the form *SendBreakWidth: value*. This line will be removed from the header before the message is sent. The default is 2000 (to allow for sending mail containing source patches).

sendWidth
When a message is sent from **xmh**, lines longer than **SendBreakWidth** characters will be split into multiple lines, each of which is no longer than this value. This value may be overridden for a single message by inserting an additional line in the message header of the form *SendWidth: value*. This line will be removed from the header before the message is sent. The default is 72.

showOnInc
Whether to automatically show the current message after incorporating new mail. Default is true.

skipCopied
Whether to skip over messages marked for copying when using ‘‘View Next Message’’ and ‘‘View Previous Message’’. Default is true.

skipDeleted
Whether to skip over messages marked for deletion when using ‘‘View Next Message’’ and ‘‘View Previous Message’’. Default is true.

skipMoved
Whether to skip over messages marked for moving to other folders when using ‘‘View Next Message’’ and ‘‘View Previous Message’’. Default is true.

stickyMenu
If true, when popup command menus are used, the most recently selected entry will be under the cursor when the menu pops up. Default is false. See the file *clients/xmh/Xmh.sample* for an example of how to specify resources for popup command menus.

tempDir
Directory for **xmh** to store temporary files. For privacy, a user might want to change this to a private directory. Default is ‘‘/tmp’’.

tocGeometry
Initial geometry for main **xmh** toc and view windows.

tocPercentage
The percentage of the main window that is used to display the Table of Contents. Default is 33.

tocWidth
How many characters to generate for each message in a folder’s table of contents. Default is 100. Use less if the geometry of the main **xmh** window results in the listing being clipped at the right hand boundary, or if you plan to use *mhl*

a lot, because it will be faster, and the extra characters may not be useful.

viewGeometry
Initial geometry for windows showing a view of a message.

MULTIPLE MAIL DROPS

Users may need to incorporate mail from multiple spool files or mail drops. If incoming mail is forwarded to the *MH slocal* program, it can be sorted as specified by the user into multiple incoming mail drops. Refer to the *MH* man page for *slocal* to learn how to specify fowarding and the automatic sorting of incoming mail in a *.maildelivery* file.

To inform **xmh** about the various mail drops, create a file in your home directory called *.xmhcheck*. In this file, a mapping between existing folder names and mail drops is created by giving a folder name followed by the absolute pathname of the mail drop site, with some white space separating them, one mapping per line. **xmh** will read this file whether or not resources are set for notification of new mail arrival, and will allow incorporation of new mail into any folder with a mail drop. **xmh** will invoke *inc* with the *–file* argument, and if **xmh** has been requested to check for new mail, it will check directly, instead of using *msgchk*.

An example of *.xmhcheck* file format, for the folders ‘‘inbox’’ and ‘‘xpert’’:

```
inbox    /var/mail/converse
xpert    /users/converse/maildrops/xpert
```

ACTIONS AND INTERFACE CUSTOMIZATION

Because **xmh** provides action procedures which correspond to command functionality and installs accelerators, users can customize accelerators and new button functionality in a private resource file. For examples of specifying customized resources, see the file *mit/clients/xmh/Xmh.sample*. To understand the syntax, see the Appendix of the *X Toolkit Intrinsics* specification on *Translation Table Syntax*, and any general explanation of using and specifying *X* resources. Unpredictable results can occur if actions are bound to events or widgets for which they were not designed.

Here’s an example of how to bind actions to your own **xmh** buttons, and how to redefine the default accelerators so that the Meta key is not required, in case you don’t have access to the sample file mentioned above.

```
! To create buttons in the middle of the main window and give them semantics:

Xmh*CommandButtonCount:              5

Xmh*commandBox.button1.label:        Inc
Xmh*commandBox.button1.translations: #override\
        <Btn1Down>,<Btn1Up>: XmhIncorporateNewMail() unset()

Xmh*commandBox.button2.label:        Compose
Xmh*commandBox.button2.translations: #override\
        <Btn1Down>,<Btn1Up>: XmhComposeMessage() unset()
```

```
Xmh*commandBox.button3.label:          Next
Xmh*commandBox.button3.translations: #override \
          <Btn1Down>,<Btn1Up>: XmhViewNextMessage() unset()

Xmh*commandBox.button4.label:          Delete
Xmh*commandBox.button4.translations: #override \
          <Btn1Down>,<Btn1Up>: XmhMarkDelete() unset()

Xmh*commandBox.button5.label:          Commit
Xmh*commandBox.button5.translations: #override \
          <Btn1Down>,<Btn1Up>: XmhCommitChanges() unset()

! To redefine the accelerator bindings to exclude modifier keys,
! and add your own keyboard accelerator for Compose Message:

Xmh*tocMenu.accelerators: #override \n \
          !:<Key>I:          XmhIncorporateNewMail() \n \
          !:<Key>C:          XmhCommitChanges() \n \
          !:<Key>R:          XmhForceRescan() \n \
          !:<Key>P:          XmhPackFolder() \n \
          !:<Key>S:          XmhSortFolder() \n
Xmh*messageMenu.accelerators: #override \n \
          !:<Key>E:          XmhComposeMessage() \n \
          !<Key>space:       XmhViewNextMessage() \n \
          !:<Key>c:          XmhMarkCopy() \n \
          !:<Key>d:          XmhMarkDelete() \n \
          !:<Key>f:          XmhForward() \n \
          !:<Key>m:          XmhMarkMove() \n \
          !:<Key>n:          XmhViewNextMessage() \n \
          !:<Key>p:          XmhViewPreviousMessage() \n \
          !:<Key>r:          XmhReply() \n \
          !:<Key>u:          XmhUnmark() \n
```

xmh provides action procedures which correspond to entries in the command menus; these are given in the sections describing menu commmands, not here. In addition to the actions corresponding to commands in the menus, these action routines are defined:

XmhPushFolder([*foldername, ...*]**)**
This action pushes each of its argument(s) onto a stack of foldernames. If no arguments are given, the selected folder is pushed onto the stack.

XmhPopFolder()
This action pops one foldername from the stack and sets the selected folder.

XmhPopupFolderMenu()
This action should always be taken when the user selects a folder button. A folder button represents a folder and zero or more subfolders. The menu of subfolders is built upon the first reference, by this routine. If there are no

subfolders, this routine will mark the folder as having no subfolders, and no menu will be built. In that case the menu button emulates a toggle button. When subfolders exist, the menu will popup, using the menu button action PopupMenu().

XmhSetCurrentFolder()
This action allows menu buttons to emulate toggle buttons in the function of selecting a folder. This action is for menu button widgets only, and sets the selected folder.

XmhLeaveFolderButton()
This action ensures that the menu button behaves properly when the user moves the pointer out of the menu button window.

XmhPushSequence([*sequencename, ...*]**)**
This action pushes each of its arguments onto the stack of sequence names. If no arguments are given, the selected sequence is pushed onto the stack.

XmhPopSequence()
This action pops one sequence name from the stack of sequence names, which then becomes the selected sequence.

XmhPromptOkayAction()
This action is equivalent to pressing the okay button in the Create Folder popup.

XmhReloadSeqLists()
This action rescans the contents of the public *MH* sequences for the currently opened folder and updates the sequence menu if necessary.

XmhShellCommand(*parameter* [, *parameter*]**)**
At least one parameter must be specified. The parameters will be concatenated with a space character separator, into a single string, and the list of selected messsages, or if no messages are selected, the current message, will be appended to the string of parameters. The string will be executed as a shell command. The messages are always given as absolute pathnames. It is an error to cause this action to execute when there are no selected messages and no current message.

XmhCheckForNewMail()
This action will check all mail drops known to xmh. If no mail drops have been specified by the user either through the *.xmhcheck* file or by the **initialIncFile** resource, the *MH* command *msgchk* is used to check for new mail, otherwise, **xmh** checks directly.

XmhWMProtocols([**wm_delete_window**] [**wm_save_yourself**])
This action is responsible for participation in window manager communication protocols. It responds to delete window and save yourself messages. The user can cause **xmh** to respond to one or both of these protocols, exactly as if the window manager had made the request, by invoking the action with the appropriate parameters. The action is insensitive to the case of the string parameters. If the event received is a ClientMessage event and parameters are

present, at least one of the parameters must correspond to the protocol requested by the event for the request to be honored by **xmh**.

CUSTOMIZATION USING *MH*

The initial text displayed in a composition window is generated by executing the corresponding *MH* command; i.e. *comp*, *repl*, or *forw*, and therefore message components may be customized as specified for those commands. *comp* is executed only once per invocation of **xmh** and the message template is re-used for every successive new composition.

xmh uses *MH* commands, including *inc*, *msgchk*, *comp*, *send*, *repl*, *forw*, *refile*, *rmm*, *pick*, *pack*, *sort*, and *scan*. Some flags for these commands can be specified in the *MH* profile; **xmh** may override them. The application resource **debug** can be set to true to see how **xmh** uses *MH* commands.

ENVIRONMENT

HOME - users's home directory
MH - to get the location of the *MH* profile file

FILES

~/.mh_profile - *MH* profile, used if the MH environment variable is not set
~/Mail - directory of folders, used if the *MH* profile cannot be found
~/.xmhcheck - optional, for multiple mail drops in cooperation with *slocal*.
/usr/local/mh6 - *MH* commands, as a last resort, see **mhPath**.
~/Mail/<folder>/.xmhcache - *scan* output in each folder
~/Mail/<folder>/.mh_sequences - sequence definitions, in each folder
/tmp - temporary files, see **tempDir**.

SEE ALSO

X11(7), **xrdb**(1), X Toolkit Intrinsics, Athena Widget Set
At least one book has been published about *MH* and **xmh**.

BUGS

- When the user closes a window, all windows which are transient for that window should also be closed by **xmh.**
- When **XmhUseAsComposition** and **XmhViewUseAsComposition** operate on messages in the **DraftsFolder**, **xmh** disallows editing of the composition if the same message is also being viewed in another window.
- Occasionally after committing changes, the table of contents will appear to be completely blank when there are actually messages present. When this happens, refreshing the display, or typing Control-L in the table of contents, will often cause the correct listing to appear. If this doesn't work, force a rescan of the folder.
- Should recognize and use the ‘‘unseen’’ message-sequence.
- Should determine by itself if the user hasn't used *MH* before, and offer to create the .mh_profile, instead of hanging on inc.
- A few commands are missing (rename folder, resend message).
- WM_DELETE_WINDOW protocol doesn't work right when requesting deletion of the first toc and view, while trying to keep other **xmh** windows around.
- Doesn't support annotations when replying to messages.
- Doesn't allow folders to be shared without write permission.

- Doesn't recognize private sequences.
- *MH* will report that the *.mh_sequences* file is poorly formatted if any sequence definition in a particular folder contains more than *BUFSIZ* characters. **xmh** tries to capture these messages and display them when they occur, but it cannot correct the problem.

COPYRIGHT

AUTHOR

Terry Weissman, formerly of Digital Western Research Laboratory
Donna Converse, MIT X Consortium

NAME

xmkmf – simple interface to the imake utility for generating X11 Makefiles

SYNOPSIS

xmkmf [**topdir** [**curdir**]]

DESCRIPTION

The **xmkmf** command runs the **imake** command with the appropriate options to generate an X11 **Makefile** from an **Imakefile** in the current directory. The X Window System uses imake extensively for both full builds of the source tree and for external software. Two variables, **TOPDIR** and **CURDIR**, can be set to make referencing files using relative path names easier. For example, the following imake command can be used to build the **Makefile** in the directory **lib/X** (relative to the top of the sources):

```
% ../.././config/imake -I../.././config \
        -DTOPDIR=../../. -DCURDIR=./lib/X
```

The above **imake** command can be similarly accomplished with the following **xmkmf** command:

```
% xmkmf ../../. ./lib/X
```

OPTIONS

topdir	Relative path to top of sources
curdir	Current directory relative to top of sources

ENVIRONMENT VARIABLES

Same as **imake**.

FILES

Imakefile	machine-independent **Makefile** descriptions

SEE ALSO

imake(1), **make**(1S)

NAME

xmodmap – utility for modifying keymaps in X

SYNOPSIS

xmodmap [-options ...] [*filename*]

DESCRIPTION

The **xmodmap** program is used to edit and display the keyboard **modifier map** and **keymap table** that are used by client applications to convert event keycodes into keysyms. It is usually run from the user's session startup script to configure the keyboard according to personal tastes.

OPTIONS

The following options may be used with **xmodmap**:

–**display** *display*
This option specifies the host and display to use.

–**help** This option indicates that a brief description of the command line arguments should be printed on the standard error channel. This will be done whenever an unhandled argument is given to **xmodmap**.

–**grammar**
This option indicates that a help message describing the expression grammar used in files and with -e expressions should be printed on the standard error.

–**verbose**
This option indicates that **xmodmap** should print logging information as it parses its input.

–**quiet** This option turns off the verbose logging. This is the default.

–**n** This option indicates that **xmodmap** should not change the mappings, but should display what it would do, like **make**(1S) does when given this option.

–**e** *expression*
This option specifies an expression to be executed. Any number of expressions may be specified from the command line.

–**pm** This option indicates that the current modifier map should be printed on the standard output.

–**pk** This option indicates that the current keymap table should be printed on the standard output.

–**pke** This option indicates that the current keymap table should be printed on the standard output in the form of expressions that can be fed back to **xmodmap**.

–**pp** This option indicates that the current pointer map should be printed on the standard output.

– A lone dash means that the standard input should be used as the input file.

The *filename* specifies a file containing **xmodmap** expressions to be executed. This file is usually kept in the user's home directory with a name like **.xmodmaprc**.

EXPRESSION GRAMMAR

The **xmodmap** program reads a list of expressions and parses them all before attempting to execute any of them. This makes it possible to refer to keysyms that are being redefined in a natural way without having to worry as much about name conflicts.

keycode *NUMBER* = *KEYSYMNAME ...*
: The list of keysyms is assigned to the indicated keycode (which may be specified in decimal, hex or octal and can be determined by running the *xev* program in the examples directory).

keysym *KEYSYMNAME* = *KEYSYMNAME ...*
: The *KEYSYMNAME* on the left hand side is translated into matching keycodes used to perform the corresponding set of **keycode** expressions. The list of keysym names may be found in the header file **<X11/keysymdef.h>** (without the **XK_** prefix) or the keysym database **/usr/openwin/lib/XKeysymDB**. Note that if the same keysym is bound to multiple keys, the expression is executed for each matching keycode.

clear *MODIFIERNAME*
: This removes all entries in the modifier map for the given modifier, where valid name are: **Shift**, **Lock**, **Control**, **Mod1**, **Mod2**, **Mod3**, **Mod4**, and **Mod5** (case does not matter in modifier names, although it does matter for all other names). For example, ‘‘clear Lock’’ will remove all any keys that were bound to the shift lock modifier.

add MODIFIERNAME = KEYSYMNAME ...
: This adds all keys containing the given keysyms to the indicated modifier map. The keysym names are evaluated after all input expressions are read to make it easy to write expressions to swap keys (see the EXAMPLES section).

remove MODIFIERNAME = KEYSYMNAME ...
: This removes all keys containing the given keysyms from the indicated modifier map. Unlike **add**, the keysym names are evaluated as the line is read in. This allows you to remove keys from a modifier without having to worry about whether or not they have been reassigned.

pointer = default
: This sets the pointer map back to its default settings (button 1 generates a code of 1, button 2 generates a 2, etc.).

pointer = *NUMBER ...*
: This sets to pointer map to contain the indicated button codes. The list always starts with the first physical button.

Lines that begin with an exclamation point (!) are taken as comments.

If you want to change the binding of a modifier key, you must also remove it from the appropriate modifier map.

EXAMPLES

Many pointers are designed such that the first button is pressed using the index finger of the right hand. People who are left-handed frequently find that it is more comfortable to reverse the button codes that get generated so that the primary button is pressed using the index finger of the left hand. This could be done on a 3 button pointer as follows:

```
% xmodmap -e "pointer = 3 2 1"
```

Many editor applications support the notion of Meta keys (similar to Control keys except that Meta is held down instead of Control). However, some servers do not have a Meta keysym in the default keymap table, so one needs to be added by hand. The following command will attach Meta to the Multi-language key (sometimes labeled Compose Character). It also takes advantage of the fact that applications that need a Meta key simply need to get the keycode and don't require the keysym to be in the first column of the keymap table. This means that applications that are looking for a Multi_key (including the default modifier map) won't notice any change.

```
% xmodmap -e "keysym Multi_key = Multi_key Meta_L"
```

One of the more simple, yet convenient, uses of **xmodmap** is to set the keyboard's "rubout" key to generate an alternate keysym. This frequently involves exchanging Backspace with Delete to be more comfortable to the user. If the *ttyModes* resource in **xterm**(1) is set as well, all terminal emulator windows will use the same key for erasing characters:

```
% xmodmap -e "keysym BackSpace = Delete"
% echo "XTerm*ttyModes: erase ˆ?" | xrdb -merge
```

Some keyboards do not automatically generate less than and greater than characters when the comma and period keys are shifted. This can be remedied with **xmodmap** by resetting the bindings for the comma and period with the following scripts:

```
!
! make shift-, be < and shift-. be >
!
keysym comma = comma less
keysym period = period greater
```

One of the more irritating differences between keyboards is the location of the Control and Shift Lock keys. A common use of **xmodmap** is to swap these two keys as follows:

```
!
! Swap Caps_Lock and Control_L
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

The **keycode** command is useful for assigning the same keysym to multiple keycodes. Although unportable, it also makes it possible to write scripts that can reset the keyboard to a known state. The following script sets the backspace key to generate Delete (as shown above), flushes all existing caps lock bindings, makes the CapsLock key be a control key, make F5 generate Escape, and makes Break/Reset be a shift lock.

```
! On the HP, the following keycodes have key caps as listed:
!
!    101  Backspace
!     55  Caps
!     14  Ctrl
!     15  Break/Reset
!     86  Stop
!     89  F5
!
keycode 101 = Delete
keycode 55 = Control_R
clear Lock
add Control = Control_R
keycode 89 = Escape
keycode 15 = Caps_Lock
add Lock = Caps_Lock
```

ENVIRONMENT

DISPLAY
to get default host and display number.

SEE ALSO

X11(7), Xlib documentation on key and pointer events

BUGS

Every time a **keycode** expression is evaluated, the server generates a **MappingNotify** event on every client. This can cause some thrashing. All of the changes should be batched together and done at once. Clients that receive keyboard input and ignore **MappingNotify** events will not notice any changes made to keyboard mappings.

Xmodmap should generate "add" and "remove" expressions automatically whenever a keycode that is already bound to a modifier is changed.

There should be a way to have the **remove** expression accept keycodes as well as keysyms for those times when you really mess up your mappings.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium, rewritten from an earlier version by David Rosenthal of Sun Microsystems.

NAME

xpr – print an X window dump

SYNOPSIS

xpr [**–device** *devtype*] [**–scale** *scale*] [**–height** *inches*] [**–width** *inches*] [**–left** *inches*] [**–top** *inches*] [**–header** *string*] [**–trailer** *string*] [**–landscape**] [**–portrait**] [**–plane** *number*] [**–gray**] [**–rv**] [**–compact**] [**–output** *filename*] [**–append** *filename*] [**–noff**] [**–split** *n*] [**–psfig**] [**–density** *dpi*] [**–cutoff** *level*] [**–noposition**] [**–gamma** *correction*] [**–render** *algorithm*] [**–slide**] [*filename*]

DESCRIPTION

xpr takes as input a window dump file produced by **xwd**(1) and formats it for output on PostScript printers, the Digital LN03 or LA100, the IBM PP3812 page printer, the HP LaserJet (or other PCL printers), or the HP PaintJet. If no file argument is given, the standard input is used. By default, **xpr** prints the largest possible representation of the window on the output page. Options allow the user to add headers and trailers, specify margins, adjust the scale and orientation, and append multiple window dumps to a single output file. Output is to standard output unless **–output** is specified.

Command Options

–device *devtype*
Specifies the device on which the file will be printed. Currently supported:

la100	Digital LA100
ljet	HP LaserJet series and other monochrome PCL devices such as ThinkJet, QuietJet, RuggedWriter, HP2560 series, and HP2930 series printers
ln03	Digital LN03
pjet	HP PaintJet (color mode)
pjetxl	HP HP PaintJet XL Color Graphics Printer (color mode)
pp	IBM PP3812
ps	PostScript printer

The default is PostScript. **-device lw** (LaserWriter) is equivalent to -device ps and is provided only for backwards compatibility.

–scale *scale*
Affects the size of the window on the page. The PostScript, LN03, and HP printers are able to translate each bit in a window pixel map into a grid of a specified size. For example each bit might translate into a 3x3 grid. This would be specified by **–scale** *3*. By default a window is printed with the largest scale that will fit onto the page for the specified orientation.

–height *inches*
Specifies the maximum height of the page.

–width *inches*
Specifies the maximum width of the page.

–left *inches*
Specifies the left margin in inches. Fractions are allowed. By default the window

is centered in the page.

–top *inches*
: Specifies the top margin for the picture in inches. Fractions are allowed.

–header *string*
: Specifies a header string to be printed above the window.

–trailer *string*
: Specifies a trailer string to be printed below the window.

–landscape
: Forces the window to printed in landscape mode. By default a window is printed such that its longest side follows the long side of the paper.

–plane *number*
: Specifies which bit plane to use in an image. The default is to use the entire image and map values into black and white based on color intensities.

–gray *2 | 3 | 4*
: Uses a simple 2x2, 3x3, or 4x4 gray scale conversion on a color image, rather than mapping to strictly black and white. This doubles, triples, or quadruples the effective width and height of the image.

–portrait
: Forces the window to be printed in portrait mode. By default a window is printed such that its longest side follows the long side of the paper.

–rv Forces the window to be printed in reverse video.

–compact
: Uses simple run-length encoding for compact representation of windows with lots of white pixels.

–output *filename*
: Specifies an output file name. If this option is not specified, standard output is used.

–append *filename*
: Specifies a filename previously produced by **xpr** to which the window is to be appended.

–noff When specified in conjunction with –**append**, the window will appear on the same page as the previous window.

–split *n* This option allows the user to split a window onto several pages. This might be necessary for very large windows that would otherwise cause the printer to overload and print the page in an obscure manner.

–psfig Suppress translation of the PostScript picture to the center of the page.

–density *dpi*
: Indicates what dot-per-inch density should be used by the HP printer.

–cutoff *level*
: Changes the intensity level where colors are mapped to either black or white for monochrome output on a LaserJet printer. The *level* is expressed as percentage of

full brightness. Fractions are allowed.

–noposition
This option causes header, trailer, and image positioning command generation to be bypassed for LaserJet, PaintJet and PaintJet XL printers.

–gamma *correction*
This changes the intensity of the colors printed by PaintJet XL printer. The *correction* is a floating point value in the range 0.00 to 3.00. Consult the operator's manual to determine the correct value for the specific printer.

–render *algorithm*
This allows PaintJet XL printer to render the image with the best quality versus performance tradeoff. Consult the operator's manual to determine which *algorithm*s are available.

–slide This option allows overhead transparencies to be printed using the PaintJet and PaintJet XL printers.

SEE ALSO

xwd(1), **xwud**(1), **X11**(7)

LIMITATIONS

The current version of **xpr** can generally print out on the LN03 most X windows that are not larger than two-thirds of the screen. For example, it will be able to print out a large Emacs window, but it will usually fail when trying to print out the entire screen. The LN03 has memory limitations that can cause it to incorrectly print very large or complex windows. The two most common errors encountered are ''band too complex'' and ''page memory exceeded.'' In the first case, a window may have a particular six pixel row that contains too many changes (from black to white to black). This will cause the printer to drop part of the line and possibly parts of the rest of the page. The printer will flash the number '1' on its front panel when this problem occurs. A possible solution to this problem is to increase the scale of the picture, or to split the picture onto two or more pages. The second problem, ''page memory exceeded,'' will occur if the picture contains too much black, or if the picture contains complex half-tones such as the background color of a display. When this problem occurs the printer will automatically split the picture into two or more pages. It may flash the number '5' on its from panel. There is no easy solution to this problem. It will probably be necessary to either cut and paste, or to rework the application to produce a less complex picture.

There are several limitations on the LA100 support: the picture will always be printed in portrait mode, there is no scaling, and the aspect ratio will be slightly off.

Support for PostScript output currently cannot handle the **-append**, **-noff** or **-split** options.

The **-compact** option is *only* supported for PostScript output. It compresses white space but not black space, so it is not useful for reverse-video windows.

For color images, should map directly to PostScript image support.

HP PRINTERS

If no –*density* is specified on the command line 300 dots per inch will be assumed for *ljet* and 90 dots per inch for *pjet*. Allowable *density* values for a LaserJet printer are 300, 150, 100, and 75 dots per inch. Consult the operator's manual to determine densities supported by other printers.

If no –**scale** is specified the image will be expanded to fit the printable page area.

The default printable page area is 8x10.5 inches. Other paper sizes can be accommodated using the –**height** and –**width** options.

Note that a 1024x768 image fits the default printable area when processed at 100 dpi with scale=1, the same image can also be printed using 300 dpi with scale=3 but will require considerably more data be transferred to the printer.

xpr may be tailored for use with monochrome PCL printers other than the LaserJet. To print on a ThinkJet (HP2225A) **xpr** could be invoked as:

```
xpr -density 96 -width 6.667 filename
```

or for black-and-white output to a PaintJet:

```
xpr -density 180 filename
```

The monochrome intensity of a pixel is computed as 0.30*R + 0.59*G + 0.11*B. If a pixel's computed intensity is less than the –**cutoff** level it will print as white. This maps light-on-dark display images to black-on-white hardcopy. The default cutoff intensity is 50% of full brightness. Example: specifying –**cutoff 87.5** moves the white / black intensity point to 87.5% of full brightness.

A LaserJet printer must be configured with sufficient memory to handle the image. For a full page at 300 dots per inch approximately 2MB of printer memory is required.

Color images are produced on the PaintJet at 90 dots per inch. The PaintJet is limited to sixteen colors from its 330 color palette on each horizontal print line. **xpr** will issue a warning message if more than sixteen colors are encountered on a line. **xpr** will program the PaintJet for the first sixteen colors encountered on each line and use the nearest matching programmed value for other colors present on the line.

Specifying the –**rv**, reverse video, option for the PaintJet will cause black and white to be interchanged on the output image. No other colors are changed.

Multiplane images must be recorded by *xwd* in *ZPixmap* format. Single plane (monochrome) images may be in either *XYPixmap* or *ZPixmap* format.

Some PCL printers do not recognize image positioning commands. Output for these printers will not be centered on the page and header and trailer strings may not appear where expected.

The –**gamma** and **-render** options are supported only on the PaintJet XL printers.

The –**slide** option is not supported for LaserJet printers.

The –**split** option is not supported for HP printers.

The –**gray** option is not supported for HP or IBM printers.

COPYRIGHT

AUTHORS

Michael R. Gretzinger, MIT Project Athena, Jose Capo, MIT Project Athena (PP3812 support), Marvin Solomon, University of Wisconsin, Bob Scheifler, MIT, Angela Bock and E. Mike Durbin, Rich Inc. (grayscale), Larry Rupp, HP (HP printer support).

NAME

xprop – property displayer for X

SYNOPSIS

xprop [-help] [-grammar] [-id *id*] [-root] [-name *name*] [-frame] [-font *font*] [-display *display*] [-len *n*] [-notype] [-fs *file*] [-remove *property-name*] [-spy] [-f *atom format* [*dformat*]]∗ [*format* [*dformat*] *atom*]∗

SUMMARY

The **xprop** utility is for displaying window and font properties in an X server. One window or font is selected using the command line arguments or possibly in the case of a window, by clicking on the desired window. A list of properties is then given, possibly with formatting information.

OPTIONS

-help Print out a summary of command line options.

-grammar
Print out a detailed grammar for all command line options.

-id *id* This argument allows the user to select window *id* on the command line rather than using the pointer to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the pointer might be impossible or interfere with the application.

-name *name*
This argument allows the user to specify that the window named *name* is the target window on the command line rather than using the pointer to select the target window.

-font *font*
This argument allows the user to specify that the properties of font *font* should be displayed.

-root This argument specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

-display *display*
This argument allows you to specify the server to connect to; see **X**(7).

-len *n* Specifies that at most *n* bytes of any property should be read or displayed.

-notype Specifies that the type of each property should not be displayed.

-fs *file* Specifies that file *file* should be used as a source of more formats for properties.

-frame Specifies that when selecting a window by hand (i.e. if none of **-name**, **-root**, or **-id** are given), look at the window manager frame (if any) instead of looking for the client window.

-remove *property-name*
Specifies the name of a property to be removed from the indicated window.

-spy Examine window properties forever, looking for property change events.

-f *name format* **[***dformat***]**
Specifies that the *format* for *name* should be *format* and that the *dformat* for *name* should be *dformat.* If *dformat* is missing, " = $0+ \ n" is assumed.

DESCRIPTION

For each of these properties, its value on the selected window or font is printed using the supplied formatting information if any. If no formatting information is supplied, internal defaults are used. If a property is not defined on the selected window or font, "not defined" is printed as the value for that property. If no property list is given, all the properties possessed by the selected window or font are printed.

A window may be selected in one of four ways. First, if the desired window is the root window, the -root argument may be used. If the desired window is not the root window, it may be selected in two ways on the command line, either by id number such as might be obtained from *xwininfo*, or by name if the window possesses a name. The -id argument selects a window by id number in either decimal or hex (must start with 0x) while the -name argument selects a window by name.

The last way to select a window does not involve the command line at all. If none of -font, -id, -name, and -root are specified, a crosshairs cursor is displayed and the user is allowed to choose any visible window by pressing any pointer button in the desired window. If it is desired to display properties of a font as opposed to a window, the -font argument must be used.

Other than the above four arguments and the -help argument for obtaining help, and the -grammar argument for listing the full grammar for the command line, all the other command line arguments are used in specifying both the format of the properties to be displayed and how to display them. The -len *n* argument specifies that at most *n* bytes of any given property will be read and displayed. This is useful for example when displaying the cut buffer on the root window which could run to several pages if displayed in full.

Normally each property name is displayed by printing first the property name then its type (if it has one) in parentheses followed by its value. The -notype argument specifies that property types should not be displayed. The -fs argument is used to specify a file containing a list of formats for properties while the -f argument is used to specify the format for one property.

The formatting information for a property actually consists of two parts, a *format* and a *dformat.* The *format* specifies the actual formatting of the property (i.e., is it made up of words, bytes, or longs?, etc.) while the *dformat* specifies how the property should be displayed.

The following paragraphs describe how to construct *format*s and *dformat*s. However, for the vast majority of users and uses, this should not be necessary as the built in defaults contain the *format*s and *dformat*s necessary to display all the standard properties. It should only be necessary to specify *format*s and *dformat*s if a new property is being dealt with or the user dislikes the standard display format. New users especially are encouraged to skip this part.

A *format* consists of one of 0, 8, 16, or 32 followed by a sequence of one or more format characters. The 0, 8, 16, or 32 specifies how many bits per field there are in the property. Zero is a special case meaning use the field size information associated with the property itself. (This is only needed for special cases like type INTEGER which is actually three different types depending on the size of the fields of the property)

A value of 8 means that the property is a sequence of bytes while a value of 16 would mean that the property is a sequence of words. The difference between these two lies in the fact that the sequence of words will be byte swapped while the sequence of bytes will not be when read by a machine of the opposite byte order of the machine that originally wrote the property. For more information on how properties are formatted and stored, consult the Xlib manual.

Once the size of the fields has been specified, it is necessary to specify the type of each field (i.e., is it an integer, a string, an atom, or what?) This is done using one format character per field. If there are more fields in the property than format characters supplied, the last character will be repeated as many times as necessary for the extra fields. The format characters and their meaning are as follows:

a	The field holds an atom number. A field of this type should be of size 32.
b	The field is an boolean. A 0 means false while anything else means true.
c	The field is an unsigned number, a cardinal.
i	The field is a signed integer.
m	The field is a set of bit flags, 1 meaning on.
s	This field and the next ones until either a 0 or the end of the property represent a sequence of bytes. This format character is only usable with a field size of 8 and is most often used to represent a string.
x	The field is a hex number (like 'c' but displayed in hex - most useful for displaying window ids and the like)

An example *format* is 32ica which is the format for a property of three fields of 32 bits each, the first holding a signed integer, the second an unsigned integer, and the third an atom.

The format of a *dformat* unlike that of a *format* is not so rigid. The only limitations on a *dformat* is that one may not start with a letter or a dash. This is so that it can be distinguished from a property name or an argument. A *dformat* is a text string containing special characters instructing that various fields be printed at various points in a manner similar to the formatting string used by printf. For example, the *dformat* " is ($0, $1 \)\n" would render the POINT 3, -4 which has a *format* of 32ii as " is (3, -4)\n".

Any character other than a $, ?, \, or a (in a *dformat* prints as itself. To print out one of $, ?, \, or (precede it by a \. For example, to print out a $, use \$. Several special backslash sequences are provided as shortcuts. \n will cause a newline to be displayed while \t will cause a tab to be displayed. *o* where *o* is an octal number will display character number *o*.

A $ followed by a number *n* causes field number *n* to be displayed. The format of the displayed field depends on the formatting character used to describe it in the corresponding *format*. I.e., if a cardinal is described by 'c' it will print in decimal while if it is described by a 'x' it is displayed in hex.

If the field is not present in the property (this is possible with some properties), <field not available> is displayed instead. $*n*+ will display field number *n* then a comma then field number *n*+1 then another comma then ... until the last field defined. If field *n* is not defined, nothing is displayed. This is useful for a property that is a list of values.

A ? is used to start a conditional expression, a kind of if-then statement. ?*exp*(*text*) will display *text* if and only if *exp* evaluates to non-zero. This is useful for two things. First, it allows fields to be displayed if and only if a flag is set. And second, it allows a value such as a state number to be displayed as a name rather than as just a number. The syntax of *exp* is as follows:

exp ::= *term* | *term*=*exp* | !*exp*

term ::= *n* | $*n* | m*n*

The ! operator is a logical ''not'', changing 0 to 1 and any non-zero value to 0. = is an equality operator. Note that internally all expressions are evaluated as 32 bit numbers so -1 is not equal to 65535. = returns 1 if the two values are equal and 0 if not. *n* represents the constant value *n* while $*n* represents the value of field number *n*. m*n* is 1 if flag number *n* in the first field having format character 'm' in the corresponding *format* is 1, 0 otherwise.

Examples: ?m3(count: $3\n) displays field 3 with a label of count if and only if flag number 3 (count starts at 0!) is on. ?$2=0(True)?!$2=0(False) displays the inverted value of field 2 as a boolean.

In order to display a property, *xprop* needs both a *format* and a *dformat*. Before *xprop* uses its default values of a *format* of 32x and a *dformat* of " = { $0+ }\n", it searches several places in an attempt to find more specific formats. First, a search is made using the name of the property. If this fails, a search is made using the type of the property. This allows type STRING to be defined with one set of formats while allowing property WM_NAME which is of type STRING to be defined with a different format. In this way, the display formats for a given type can be overridden for specific properties.

The locations searched are in order: the format if any specified with the property name (as in 8x WM_NAME), the formats defined by -f options in last to first order, the contents of the file specified by the -fs option if any, the contents of the file specified by the environmental variable XPROPFORMATS if any, and finally *xprop*'s built in file of formats.

The format of the files referred to by the -fs argument and the XPROPFORMATS variable is one or more lines of the following form:

name format [*dformat*]

Where *name* is either the name of a property or the name of a type, *format* is the *format* to be used with *name* and *dformat* is the *dformat* to be used with *name*. If *dformat* is not present, " = $0+\n" is assumed.

EXAMPLES

To display the name of the root window: *xprop* -root WM_NAME

To display the window manager hints for the clock: *xprop* -name xclock WM_HINTS

To display the start of the cut buffer: *xprop* -root -len 100 CUT_BUFFER0

To display the point size of the fixed font: *xprop* -font fixed POINT_SIZE

To display all the properties of window # 0x200007: *xprop* -id 0x200007

ENVIRONMENT

DISPLAY
To get default display.

XPROPFORMATS
Specifies the name of a file from which additional formats are to be obtained.

SEE ALSO

X11(7), **xwininfo**(1)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X**(7) for a full statement of rights and permissions.

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xrdb – X server resource database utility

SYNOPSIS

xrdb [-option ...] [*filename*]

DESCRIPTION

Xrdb is used to get or set the contents of the RESOURCE_MANAGER property on the root window of screen 0, or the SCREEN_RESOURCES property on the root window of any or all screens, or everything combined. You would normally run this program from your X startup file.

Most X clients use the RESOURCE_MANAGER and SCREEN_RESOURCES properties to get user preferences about color, fonts, and so on for applications. Having this information in the server (where it is available to all clients) instead of on disk, solves the problem in previous versions of X that required you to maintain *defaults* files on every machine that you might use. It also allows for dynamic changing of defaults without editing files.

The RESOURCE_MANAGER property is used for resources that apply to all screens of the display. The SCREEN_RESOURCES property on each screen specifies additional (or overriding) resources to be used for that screen. (When there is only one screen, SCREEN_RESOURCES is normally not used, all resources are just placed in the RESOURCE_MANAGER property.)

The file specified by *filename* (or the contents from standard input if - or no filename is given) is optionally passed through the C preprocessor with the following symbols defined, based on the capabilities of the server being used:

BITS_PER_RGB=num
the number of significant bits in an RGB color specification. This is the log base 2 of the number of distinct shades of each primary that the hardware can generate. Note that it usually is not related to PLANES.

CLASS=visualclass
one of StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, DirectColor. This is the visual class of the root window of the default screen.

COLOR
defined only if CLASS is one of StaticColor, PseudoColor, TrueColor, or DirectColor.

HEIGHT=num
the height of the default screen in pixels.

SERVERHOST=hostname
the hostname portion of the display to which you are connected.

HOST=hostname
the same as **SERVERHOST**.

CLIENTHOST=hostname
the name of the host on which **xrdb** is running.

PLANES=num

the number of bit planes (the depth) of the root window of the default screen.

RELEASE=num
: the vendor release number for the server. The interpretation of this number will vary depending on VENDOR.

REVISION=num
: the X protocol minor version supported by this server (currently 0).

VERSION=num
: the X protocol major version supported by this server (should always be 11).

VENDOR=vendor
: a string specifying the vendor of the server.

WIDTH=num
: the width of the default screen in pixels.

X_RESOLUTION=num
: the x resolution of the default screen in pixels per meter.

Y_RESOLUTION=num
: the y resolution of the default screen in pixels per meter.

Lines that begin with an exclamation mark (!) are ignored and may be used as comments.

Note that since **xrdb** can read from standard input, it can be used to the change the contents of properties directly from a terminal or from a shell script.

OPTIONS

xrdb program accepts the following options:

–help This option (or any unsupported option) will cause a brief description of the allowable options and parameters to be printed.

–display *display*
: This option specifies the X server to be used; see **X**(7). It also specifies the screen to use for the *-screen* option, and it specifies the screen from which preprocessor symbols are derived for the *-global* option.

–all This option indicates that operation should be performed on the screen-independent resource property (RESOURCE_MANAGER), as well as the screen-specific property (SCREEN_RESOURCES) on every screen of the display. For example, when used in conjunction with *-query*, the contents of all properties are output. For *-load* and *-merge*, the input file is processed once for each screen. The resources which occur in common in the output for every screen are collected, and these are applied as the screen-independent resources. The remaining resources are applied for each individual per-screen property. This the default mode of operation.

–global This option indicates that the operation should only be performed on the screen-independent RESOURCE_MANAGER property.

–screen This option indicates that the operation should only be performed on the SCREEN_RESOURCES property of the default screen of the display.

–screens

This option indicates that the operation should be performed on the SCREEN_RESOURCES property of each screen of the display. For *-load* and *-merge*, the input file is processed for each screen.

–**n** This option indicates that changes to the specified properties (when used with *-load* or *-merge*) or to the resource file (when used with *-edit*) should be shown on the standard output, but should not be performed.

–**quiet** This option indicates that warning about duplicate entries should not be displayed.

-cpp *filename*
This option specifies the pathname of the C preprocessor program to be used. Although **xrdb** was designed to use CPP, any program that acts as a filter and accepts the -D, -I, and -U options may be used.

-nocpp This option indicates that **xrdb** should not run the input file through a preprocessor before loading it into properties.

–**symbols**
This option indicates that the symbols that are defined for the preprocessor should be printed onto the standard output.

–**query** This option indicates that the current contents of the specified properties should be printed onto the standard output. Note that since preprocessor commands in the input resource file are part of the input file, not part of the property, they won't appear in the output from this option. The –**edit** option can be used to merge the contents of properties back into the input resource file without damaging preprocessor commands.

–**load** This option indicates that the input should be loaded as the new value of the specified properties, replacing whatever was there (i.e. the old contents are removed). This is the default action.

–**merge** This option indicates that the input should be merged with, instead of replacing, the current contents of the specified properties. Note that this option does a lexicographic sorted merge of the two inputs, which is almost certainly not what you want, but remains for backward compatibility.

–**remove**
This option indicates that the specified properties should be removed from the server.

–**retain** This option indicates that the server should be instructed not to reset if **xrdb** is the first client. This never be necessary under normal conditions, since **xdm**(1) and **xinit**(1) always act as the first client.

–**edit** *filename*
This option indicates that the contents of the specified properties should be edited into the given file, replacing any values already listed there. This allows you to put changes that you have made to your defaults back into your resource file, preserving any comments or preprocessor lines.

–**backup** *string*

This option specifies a suffix to be appended to the filename used with –**edit** to generate a backup file.

–**D***name[=value]*
This option is passed through to the preprocessor and is used to define symbols for use with conditionals such as *#ifdef.*

–**U***name* This option is passed through to the preprocessor and is used to remove any definitions of this symbol.

–**I***directory*
This option is passed through to the preprocessor and is used to specify a directory to search for files that are referenced with *#include.*

FILES

Generalizes *~/.Xdefaults* files.

SEE ALSO

X11(7), *Xlib Resource Manager* documentation, Xt resource documentation

ENVIRONMENT

DISPLAY
to figure out which display to use.

BUGS

The default for no arguments should be to query, not to overwrite, so that it is consistent with other programs.

WARNINGS

By default, **xrdb** passes its input through the **cpp**(1) pre-processor. The **cpp** pre-processor defines the symbol "sun" with the value "1". This means that any unquoted appearance of the string "sun" in the input will be converted to the string "1". To avoid this behavior, invoke the command with the option **-nocpp** or the option **-Usun .** Alternatively, precede a group of lines that contain the string "sun" with the line **#undef sun** and follow them by **#define sun 1**

COPYRIGHT

Copyright 1991, Digital Equipment Corporation and MIT.

AUTHORS

Bob Scheifler, Phil Karlton, rewritten from the original by Jim Gettys

NAME

XReadScreen – returns the displayed colors in a rectangle of the screen

SYNOPSIS

XImage
XReadScreen (Display *∗display*, **Window** *w*, **int** *x*, **int** *y*, **unsigned int** *width*, **unsigned int** *height*, **Bool** *includeCursor)*

Arguments

display Specifies the connection to the X server.

w Specifies the window from whose screen the data is read.

x, *y* Specify the X and Y coordinates of the upper-left corner of the rectangle relative to the origin of the window *w*.

width, height
Specify the width and height of the rectangle.

includeCursor
Specifies whether the cursor image is to be included in the colors returned.

DESCRIPTION

This routine provides access to the colors displayed on the screen of the given window. On some types of advanced display devices, the displayed colors can be a composite of the data contained in several different frame stores and these frame stores can be of different depth and visual types.

In addition, there can be overlay / underlay window pairs in which part of the underlay is visible beneath the overlay. Because the data returned by **XGetImage** is undefined for portions of the rectangle that have different depths, **XGetImage** is inadequate to return a picture of the what user is actually seeing on the screen. In addition, **XGetImage** cannot composite pixel information for an overlay / underlay window pair because the pixel information lies in different drawables. **XReadScreen** addresses these problems.

Rather than returning pixel information, **XReadScreen** returns color information-the actual displayed colors visible on the screen. It returns the color information from any window within the boundaries of the specified rectangle. Unlike **XGetImage,** the returned contents of visible regions of inferior or overlapping windows of a different depth than the specified window's depth are not undefined. Instead, the actual displayed colors for these windows is returned.

Note: The colors returned are the ones that would be displayed if an unlimited number of hardware color LUTs were available on the screen. Thus, the colors returned are the theoretical display colors. If colormap flashing is present on the screen because there aren't enough hardware color LUTs to display all of the software colormaps simultaneously, the returned colors may be different from the colors that are actually displayed.

If *w* is an overlay window, the overlay color information is returned everywhere there is opaque paint in the specified rectangle. The color information of the underlay is returned everywhere there is transparent paint in the overlay. In general, since this underlay can be an overlay window containing transparent paint, the color information for a coordinate (x, y) which contains transparent paint is the youngest non-inferior that has opaque paint at (x, y).

The color data is returned as an **XImage.** The returned image has the same width and height as the arguments specified. The format of the image is **ZPixmap.** The depth of the image is 24 and the bits_per_pixel is 32. The most significant 8 bits of color information for each color channel (red, green blue) will be returned in the bit positions defined by *red_mask, green_mask,* and *blue_mask* in the **XImage.** The values of the following attributes of the **XImage** are server dependent: *byte_order, bitmap_unit, bitmap_bit_order, bitmap_pad, bytes_per_line, red_mask, green_mask, blue_mask.*

If *includeCursor* is **True,** the cursor image is included in the returned colors. Otherwise, it is excluded.

Note that the borders of the argument window (and other windows) can be included and read with this request.

If a problem occurs, **XReadScreen** returns NULL.

NAME

xrefresh – refresh all or part of an X screen

SYNOPSIS

xrefresh [**–white**] [**–black**] [**–solid** *color*] [**–root**] [**–none**] [**–geometry** *WxH+X+Y*] [**–display** *display*]

DESCRIPTION

Xrefresh is a simple X program that causes all or part of your screen to be repainted. This is useful when system messages have messed up your screen. **Xrefresh** maps a window on top of the desired area of the screen and then immediately unmaps it, causing refresh events to be sent to all applications. By default, a window with no background is used, causing all applications to repaint ''smoothly.'' However, the various options can be used to indicate that a solid background (of any color) or the root window background should be used instead.

OPTIONS

–white Use a white background. The screen just appears to flash quickly, and then repaint.

–black Use a black background (in effect, turning off all of the electron guns to the tube). This can be somewhat disorienting as everything goes black for a moment.

–solid *color*
Use a solid background of the specified color. Try green.

–root Use the root window background.

–none This is the default. All of the windows simply repaint.

–geometry *WxH+X+Y*
Specifies the portion of the screen to be repainted; see **X**(7).

–display *display*
This argument allows you to specify the server and screen to refresh; see **X**(7).

X DEFAULTS

The **xrefresh** program uses the routine **XGetDefault**(3X) to read defaults, so its resource names are all capitalized.

Black, **White**, **Solid**, **None**, **Root**
Determines what sort of window background to use.

Geometry
Determines the area to refresh. Not very useful.

ENVIRONMENT

DISPLAY - To get default host and display number.

SEE ALSO

X11(7)

BUGS

It should have just one default type for the background.

COPYRIGHT

AUTHORS

Jim Gettys, Digital Equipment Corp., MIT Project Athena

NAME

xscope – X Window Protocol Viewer

SYNOPSIS

xscope [**-i** *input-port*] [**-o** *output-port*] [**-h** *host*] [**-d** *display*] [**-q**] [**-v** *print-level*]

DESCRIPTION

Xscope sits in-between an X11 client and an X11 server and prints the contents of each request, reply, error, or event that is communicated between them. This information can be useful in debugging and performance tuning of X11 servers and clients.

To operate, **xscope** must know the host, port, and display to use to connect to the X11 server. In addition, it must know the port on which it should listen for X11 clients. Two cases are common:

(1) The X11 server is on the same host as **xscope.** In this case, the input port for **xscope** should be selected as an X11 server on a different display, and the client DISPLAY argument adjusted to select **xscope.** For example, if the X11 server is on port 6000, display 0, then **xscope** can use port 6001 as its input port. The client can use display 0 for direct access to X11 or display 1 for access to **xscope.**

(2) The X11 server is on a different host than **xscope.** In this case the same input and output ports can be used, and the host component of the DISPLAY is used to select **xscope** or X11.

OPTIONS

-i *input-port*
Specify the port that **xscope** will use to take requests from clients (defaults to 1). For X11, this port is automatically biased by 6000.

-o *output-port*
Determines the port that **xscope** will use to connect to X11 (defaults to 0). For X11, this port is automatically biased by 6000.

-h *host* Determines the host that **xscope** will use to find its X11 server.

-d *display* Defines the display number. The display number is added to the input and output port to give the actual ports which are used by **xscope.**

-q Quiet output mode. Gives only the names of requests, replies, errors, and events, but does not indicate contents.

-v *print-level*
Determines the level of printing which **xscope** will provide. The print-level can be 0 (same as quiet mode), 1, 2, 3, 4. The larger numbers give more and more output. For example, a successful setup returns a string which is the name of the vendor of the X11 server. At level 1, the explicit field giving the length of the string is suppressed since it can be inferred from the string. At level 2 and above the length is explicitly printed.

EXAMPLES

```
xscope -i1 -o0 < /dev/null >& /tmp/xscope.out & client -display localhost:1
```

This command would have **xscope** communicate with an X11 server on the local host, display 0; xscope itself would be available on the current host as display 1 (display of 0 plus the 1 of -i1). The standard input is redirected from /dev/null to prevent **xscope** from stopping when put into the background. Output is redirected to a file in /tmp.

xscope -v4 -hcleo -d0 -o0 -i1

This command would have **xscope** communicate with an X11 server on host ‘‘cleo’’, display 0; xscope itself would be available on the current host as display 1 (display of 0 plus the 1 of -i1). Verbose level 4.

xscope -q -d1 -o1 -o3

The X11 server for the current host, display 2 (1 for -d1 plus 1 for -o1) would be used by **xscope** which would run as display 4 (1 for -d1 plus 3 for -o3). Quiet mode (verbose level 0).

SEE ALSO

X11(7)

AUTHOR

James L. Peterson (MCC)

NAME

Xserver – X Window System server

SYNOPSIS

X [:displaynumber] [–option ...] [ttyname]

DESCRIPTION

X is the generic name for the X Window System server. It is frequently a link or a copy of the appropriate server binary for driving the most frequently used server on a given machine.

STARTING THE SERVER

The server is usually started from the X Display Manager program **xdm**(1). This utility is run from the system boot files and takes care of keeping the server running, prompting for usernames and passwords, and starting up the user sessions. It is easily configured for sites that wish to provide nice, consistent interfaces for novice users (loading convenient sets of resources, starting up a window manager, clock, and nice selection of terminal emulator windows).

Installations that run more than one window system will still need to use the **xinit**(1) utility. However, **xinit** is to be considered a tool for building startup scripts and is not intended for use by end users. Site administrators are **strongly** urged to use **xdm**, or build other interfaces for novice users.

When the X server starts up, it takes over the display. If you are running on a workstation whose console is the display, you cannot log into the console while the server is running.

NETWORK CONNECTIONS

The X server supports connections made using the following reliable byte-streams:

TCP/IP
Listen on port 6000+*n*, where *n* is the display number.

Unix Domain
Use **/tmp/.X11-unix/X***n* as the filename for the socket, where *n* is the display number.

DECnet
Respond to connections to object **X$X***n*, where *n* is the display number. This is not supported in all environments.

OPTIONS

All X servers accept the following command line options:

–a *number*
Set pointer acceleration (i.e. the ratio of how much is reported to how much the user actually moved the pointer). The default is 2.

–ac Disable host-based access control mechanisms. Enables access by any host, and permits any host to modify the access control list. Use with caution. This option exists primarily for executing remote test suites.

–audit *level*
Set the audit trail level. The default is 1; only connection rejections are reported. Level 2 also reports successful connections and disconnects. Level 0 turns off the audit trail. Audit output is sent on the standard error output stream.

–auth *authorization-file*
Specify a file which contains a collection of authorization records used to authenticate access. See also **xdm**(1)

bc Disable certain kinds of error checking for bug compatibility with previous releases (e.g., to work around bugs in R2 and R3 xterms and toolkits). Deprecated.

–bs Disable backing store support on all screens.

–c Turn off key-click.

c *volume* Set key-click volume, range is 0-100.

–cc *class*
Set the visual class for the root window of color screens. Class numbers are as specified in the X protocol. Not implemented in all servers.

–co *filename*
Set name of RGB color database.

–core The server dumps core on fatal errors.

–dpi *resolution*
Set the resolution of the screen, in dots per inch. Use when the server cannot determine the screen size from the hardware. The default is 90.

–f *volume*
Set beep (bell) volume, range is 0-100. The default is 50.

–fc *cursorFont*
Set default cursor font.

–fn *font* Set default font.

–fp *fontPath*
Set the font search path. This path is a comma separated list of directories which the X server uses to search for font databases.

–ep *encodingPath*
Set the font encoding search path.

–help Print a usage message.

–I All following command line arguments are ignored.

–logo Turn on the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

nologo Turn off the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

–p *minutes*
Set screen-saver pattern cycle time in minutes. The default is 10 minutes.

–pn The server is to continue running if it fails to establish all of its well-known sockets, but establishes at least one.

–r Turn off auto-repeat.

r Turn on auto-repeat.

–s *minutes*
Set screen-saver timeout time in minutes. The default is 10 minutes.

–su Disable save under support on all screens.

–t *number*
Set pointer acceleration threshold in pixels (i.e. after how many pixels pointer acceleration should take effect). The default is 4.

–terminate
The server is to terminate instead of resetting.

–to *seconds*
Set default connection timeout in seconds. The default is 60 seconds.

–tst Disable all testing extensions (e.g., XTEST, XTrap, XTestExtension1).

tty*xx* ignored, for servers started the ancient way (from init).

v Set video-off screen-saver preference.

–v Set video-on screen-saver preference.

–wm Force the default backing-store of all windows to be WhenMapped; an inexpensive way of getting backing-store to apply to all windows.

–x *extension*
Load the specified extension at init. Not supported in most implementations.

You can also have the X server connect to **xdm**(1) using XDMCP. Although this is not typically useful as it does not allow **xdm** to manage the server process, it can be used to debug XDMCP implementations, and serves as a sample implementation of the server side of XDMCP. For more information on this protocol, see the *X Display Manager Control Protocol* specification. The following options control the behavior of XDMCP.

–query *host-name*
Enable XDMCP and send Query packets to the specified host.

–broadcast
Enable XDMCP and broadcast BroadcastQuery packets to the network. The first responding display manager will be chosen for the session.

–indirect *host-name*
Enable XDMCP and send IndirectQuery packets to the specified host.

–port *port-num*
Use an alternate port number for XDMCP packets. Must be specified before any –query, –broadcast or –indirect options.

–once Terminate the server after one session.

–class *display-class*
XDMCP has an additional display qualifier used in resource lookup for display-specific options. This option sets that value, by default it is "MIT-Unspecified" (not a very useful value).

–displayID *display-id*

Yet another XDMCP specific value, this one allows the display manager to identify each display so that it can locate the shared key.

Many servers also have device-specific command line options. See the manual pages for the individual servers for more details.

SECURITY

The X server implements a simple authorization protocol, MIT-MAGIC-COOKIE-1 which uses data private to authorized clients and the server. This is a rather trivial scheme; if the client passes authorization data which is the same as that held by the server, it is allowed connection access. This scheme is worse than the host-based access control mechanisms in environments with unsecure networks as it allows any host to connect, given that it has the private key. But in many environments, this level of security is better than the host-based scheme as it provides access control on a per-user instead of per-host basis.

In addition, the server provides support for a DES-based authorization scheme, SUN-DES-1, using Sun's Secure RPC. It involves encrypting data with the X server's public key. See the *Solaris X Window System Developers Guide* for more information.

The authorization data is passed to the server in a private file named with the −**auth** command line option. Before accepting connections after a reset or when the server is starting, it reads this file. If this file contains authorization records, the local host is not allowed access to the server; only clients which send one of the authorization records contained in the file in the connection setup information are allowed access. See the **xauth**(1) manual page for a description of the binary format of this file.

The X server also uses a host-based access control list for deciding whether or not to accept connections from clients on a particular machine. If no other authorization mechanism is being used, this list initially consists of the host on which the server is running as well as any machines listed in the file **/etc/X***n***.hosts**, where *n* is the display number of the server. The file contains either an Internet hostname (e.g. expo.lcs.mit.edu) or a DECnet hostname in double colon format (e.g. hydra::). Each hostname must be newline separated with no leading or trailing whitespace. For example:

```
joesworkstation
corporate.company.com
star::
bigcpu::
```

Users add or remove hosts from this list and enable or disable access control using the **xhost** command from the same machine as the server.

The X protocol intrinsically does not have any notion of window operation permissions or place any restrictions on what a client can do; if a program can connect to a display, it has full run of the screen. Sites that have better authentication and authorization systems (such as Kerberos) might wish to make use of the hooks in the libraries and the server to provide additional security models.

SIGNALS

The X server handles the following signals:

SIGHUP
: Close all existing connections, free all resources, and restore server defaults. This signal is sent by the display manager whenever the user's primary application (usually an **xterm**(1) or window manager) exits to force the server to clean up and prepare for the next user.

SIGTERM
: The server exits cleanly.

SIGUSR1
: This signal is used quite differently from either of the above. When the server starts, determines if SIGUSR1 is set to SIG_IGN. If this is true the server sends a SIGUSR1 to its parent process after it has set up the various connection schemes. **Xdm** uses this feature to recognize when the server is ready for client connections (see **xdm**(1)).

FONTS

Fonts are normally stored as individual files across various directories. The X server can obtain fonts from directories and / or from font servers. The list of directories and font servers the X server uses when trying to open a font is controlled by the *font path.* Although most sites will choose to have the X server start up with the appropriate font path (see the −**fp**), This path can be overridden using the **xset**(1) program.

The default font path for the X server includes eight directories:

/usr/openwin/lib/X11/fonts/misc
: This directory contains many miscellaneous bitmap fonts that are useful on all systems. It contains a family of fixed-width fonts, a family of fixed-width fonts from Dale Schumacher, several Kana fonts from Sony Corporation, two JIS Kanji fonts, two Hangul fonts from Daewoo Electronics, two Hebrew fonts from Joseph Friedman, the standard cursor font, two cursor fonts from Digital Equipment Corporation, and cursor and glyph fonts from Sun Microsystems. It also has various font name aliases for the fonts, including **fixed** and **variable**.

/usr/openwin/lib/X11/fonts/Speedo
: This directory contains outline fonts for Bitstream's Speedo rasterizer. A single font face, contributed by Bitstream, Inc., in normal, bold, italic, and bold italic, is provided.

/usr/openwin/lib/X11/fonts/75dpi
: This directory contains bitmap fonts contributed by Adobe Systems, Inc., Digital Equipment Corporation, Bitstream, Inc., Bigelow and Holmes, and Sun Microsystems, Inc. for 75 dots per inch displays. An integrated selection of sizes, styles, and weights are provided for each family.

/usr/openwin/lib/X11/fonts/100dpi
: This directory contains 100 dots per inch versions of some of the fonts in the *75dpi* directory.

/usr/openwin/lib/X11/fonts/F3
: This directory contains scalable outlie fonts for the F3 format. 57 typefaces are

present.

/usr/openwin/lib/X11/fonts/F3bitmaps
This directory contains bitmaps in various point sized for the 57 F3 fonts.

/usr/openwin/lib/X11/fonts/Xt+
This directory contains fonts used by OLIT.

/usr/openwin/lib/X11/fonts/Type1
This directory contains outline Adobe Type1 fonts.

Font databases are created by executing the **mkfontdir**(1) program in the directory containing the compiled versions of the fonts (the **.pcf** files). Whenever fonts are added to a directory, **mkfontdir** should be executed so that the server can find the new fonts. If **mkfontdir** is not run, the server will not find any fonts in the directory.

DIAGNOSTICS

Too numerous to list them all. If run from **init**(1M), errors are typically logged in the file **/usr/adm/X*msgs**,

FILES

/etc/X*.hosts	Initial access control list
/usr/openwin/lib/X11/fonts/misc	Bitmap font directory
/usr/openwin/lib/X11/fonts/75dpi	Bitmap font directory
/usr/openwin/lib/X11/fonts/100dpi	Bitmap font directory
/usr/openwin/lib/X11/fonts/Speedo	Outline font directory
/usr/openwin/lib/X11/fonts/F3	F3 outline font directory
/usr/openwin/lib/X11/fonts/F3bitmaps	Bitmap font directory
/usr/openwin/lib/X11/fonts/Xt+	OLIT font directory
/usr/openwin/lib/X11/fonts/Type1	Outline font directories
/usr/openwin/lib/X11/fonts/PEX	PEX font directories
/usr/openwin/lib/X11/rgb.txt	Color database
/tmp/.X11-unix/X*	Unix domain socket
/usr/adm/X*msgs	Error log file

SEE ALSO

X11(7), **bdftopcf**(1), **mkfontdir**(1), **xauth**(1), **xdm**(1), **xhost**(1), **xinit**(1), **xset**(1), **xsetroot**(1), **xterm**(1), **Xsun**(1) *X Window System Protocol, Definition of the Porting Layer for the X v11 Sample Server, Strategies for Porting the X v11 Sample Server, Godzilla's Guide to*

Porting the X V11 Sample Server

BUGS

The option syntax is inconsistent with itself and **xset**(1).

The acceleration option should take a numerator and a denominator like the protocol.

If **X** dies before its clients, new clients won't be able to connect until all existing connections have their TCP TIME_WAIT timers expire.

The color database is missing a large number of colors.

COPYRIGHT

AUTHORS

The sample server was originally written by Susan Angebranndt, Raymond Drewry, Philip Karlton, and Todd Newman, from Digital Equipment Corporation, with support from a large cast. It has since been extensively rewritten by Keith Packard and Bob Scheifler, from MIT.

NAME

xset – user preference utility for X

SYNOPSIS

xset [-display *display*] [-b] [b on / off] [b [*volume* [*pitch* [*duration*]]] [[-]bc] [-c] [c on / off] [c [*volume*]] [[-+]fp[-+=] *path*[,*path*[,...]]] [fp default] [fp rehash] [[-]led [*integer*]] [led on / off] [m[ouse] [*accel_mult*[/ *accel_div*] [*threshold*]]] [m[ouse] default] [p *pixel color*] [[-]r] [r on / off] [s [*length* [*period*]]] [s blank / noblank] [s expose / noexpose] [s on / off] [s default] [q]

DESCRIPTION

This program is used to set various user preference options of the display.

OPTIONS

–display *display*
This option specifies the server to use; see *X(1)*.

b The **b** option controls bell volume, pitch and duration. This option accepts up to three numerical parameters, a preceding dash(-), or a 'on / off' flag. If no parameters are given, or the 'on' flag is used, the system defaults will be used. If the dash or 'off' are given, the bell will be turned off. If only one numerical parameter is given, the bell volume will be set to that value, as a percentage of its maximum. Likewise, the second numerical parameter specifies the bell pitch, in hertz, and the third numerical parameter specifies the duration in milliseconds. Note that not all hardware can vary the bell characteristics. The X server will set the characteristics of the bell as closely as it can to the user's specifications.

bc The *bc* option controls *bug compatibility* mode in the server, if possible; a preceding dash(-) disables the mode, otherwise the mode is enabled. Various pre-R4 clients pass illegal values in some protocol requests, and pre-R4 servers did not correctly generate errors in these cases. Such clients, when run against an R4 server, will terminate abnormally or otherwise fail to operate correctly. Bug compatibility mode explicitly reintroduces certain bugs into the X server, so that many such clients can still be run. This mode should be used with care; new application development should be done with this mode disabled. The server must support the MIT-SUNDRY-NONSTANDARD protocol extension in order for this option to work.

c The **c** option controls key click. This option can take an optional value, a preceding dash(-), or an 'on / off' flag. If no parameter or the 'on' flag is given, the system defaults will be used. If the dash or 'off' flag is used, keyclick will be disabled. If a value from 0 to 100 is given, it is used to indicate volume, as a percentage of the maximum. The X server will set the volume to the nearest value that the hardware can support.

fp= *path,...*
The **fp=** sets the font path to the directories given in the path argument. The directories are interpreted by the server, not by the client, and are server-dependent. Directories that do not contain font databases created by *mkfontdir* will be ignored by the server.

fp default

The **default** argument causes the font path to be reset to the server's default.

fp rehash
The **rehash** argument causes the server to reread the font databases in the current font path. This is generally only used when adding new fonts to a font directory (after running *mkfontdir* to recreate the font database).

–fp or **fp–**
The **–fp** and **fp–** options remove elements from the current font path. They must be followed by a comma-separated list of directories.

+fp or **fp+**
This **+fp** and **fp+** options prepend and append elements to the current font path, respectively. They must be followed by a comma-separated list of directories.

led The **led** option controls the keyboard LEDs. This controls the turning on or off of one or all of the LEDs. It accepts an optional integer, a preceding dash(-) or an 'on/off' flag. If no parameter or the 'on' flag is given, all LEDs are turned on. If a preceding dash or the flag 'off' is given, all LEDs are turned off. If a value between 1 and 32 is given, that LED will be turned on or off depending on the existence of a preceding dash. A common LED which can be controlled is the ''Caps Lock'' LED. ''xset led 4'' would turn led #4 on. ''xset -led 4'' would turn it off. The particular LED values may refer to different LEDs on different hardware.

m The **m** option controls the mouse parameters. The parameters for the mouse are 'acceleration' and 'threshold'. The acceleration can be specified as an integer, or as a simple fraction. The mouse, or whatever pointer the machine is connected to, will go 'acceleration' times as fast when it travels more than 'threshold' pixels in a short time. This way, the mouse can be used for precise alignment when it is moved slowly, yet it can be set to travel across the screen in a flick of the wrist when desired. One or both parameters for the **m** option can be omitted, but if only one is given, it will be interpreted as the acceleration. If no parameters or the flag 'default' is used, the system defaults will be set.

p The **p** option controls pixel color values. The parameters are the color map entry number in decimal, and a color specification. The root background colors may be changed on some servers by altering the entries for BlackPixel and WhitePixel. Although these are often 0 and 1, they need not be. Also, a server may choose to allocate those colors privately, in which case an error will be generated. The map entry must not be a read-only color, or an error will result.

r The **r** option controls the autorepeat. If a preceding dash or the 'off' flag is used, autorepeat will be disabled. If no parameters or the 'on' flag is used, autorepeat will be enabled.

s The **s** option lets you set the screen saver parameters. This option accepts up to two numerical parameters, a 'blank/noblank' flag, an 'expose/noexpose' flag, an 'on/off' flag, or the 'default' flag. If no parameters or the 'default' flag is used, the system will be set to its default screen saver characteristics. The 'on/off' flags simply turn the screen saver functions on or off. The 'blank' flag sets the preference to blank the video (if the hardware can do so) rather than display a background pattern, while 'noblank' sets the preference to display a pattern rather than blank the video. The 'expose' flag sets the preference to allow window exposures (the server can freely discard window contents), while 'noexpose' sets the preference to disable screen saver unless the server can regenerate the screens without causing exposure events. The length and period parameters for the screen saver function determines how long the server must be inactive for screen saving to activate, and the period to change the background pattern to avoid burn in. The arguments are specified in seconds. If only one numerical parameter is given, it will be used for the length.

q The **q** option gives you information on the current settings.

These settings will be reset to default values when you log out.

Note that not all X implementations are guaranteed to honor all of these options.

SEE ALSO

X11(7) **Xserver**(1), **xmodmap**(1), **xrdb**(1), **xsetroot**(1)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science
David Krikorian, MIT Project Athena (X11 version)

NAME

xsetroot – root window parameter setting utility for X

SYNOPSIS

xsetroot [-help] [-def] [-display *display*] [-cursor *cursorfile maskfile*] [-cursor_name *cursor-name*] [-bitmap *filename*] [-mod *x y*] [-gray] [-grey] [-fg *color*] [-bg *color*] [-rv] [-solid *color*] [-name *string*]

DESCRIPTION

The **setroot** program allows you to tailor the appearance of the background ("root") window on a workstation display running X. Normally, you experiment with **xsetroot** until you find a personalized look that you like, then put the **xsetroot** command that produces it into your X startup file. If no options are specified, or if *-def* is specified, the window is reset to its default state. The *-def* option can be specified along with other options and only the non-specified characteristics will be reset to the default state.

Only one of the background color / tiling changing options (-solid, -gray, -grey, -bitmap, and -mod) may be specified at a time.

OPTIONS

The various options are as follows:

-help Print a usage message and exit.

-def Reset unspecified attributes to the default values. (Restores the background to the familiar gray mesh and the cursor to the hollow x shape.)

-cursor *cursorfile maskfile*
This lets you change the pointer cursor to whatever you want when the pointer cursor is outside of any window. Cursor and mask files are bitmaps (little pictures), and can be made with the **bitmap**(1) program. You probably want the mask file to be all black until you get used to the way masks work.

-cursor_name *cursorname*
This lets you change the pointer cursor to one of the standard cursors from the cursor font. Refer to appendix B of the X protocol for the names (except that the XC_ prefix is elided for this option).

-bitmap *filename*
Use the bitmap specified in the file to set the window pattern. You can make your own bitmap files (little pictures) using the *bitmap(1)* program. The entire background will be made up of repeated "tiles" of the bitmap.

-mod *x y*
This is used if you want a plaid-like grid pattern on your screen. x and y are integers ranging from 1 to 16. Try the different combinations. Zero and negative numbers are taken as 1.

-gray Make the entire background gray. (Easier on the eyes.)

-grey Make the entire background grey.

-fg *color*
Use ‘‘color’’ as the foreground color. Foreground and background colors are meaningful only in combination with -cursor, -bitmap, or -mod.

-bg *color*

Use ‘‘color’’ as the background color.

-rv This exchanges the foreground and background colors. Normally the foreground color is black and the background color is white.

-solid *color*
This sets the background of the root window to the specified color. This option is only useful on color servers.

-name *string*
Set the name of the root window to ‘‘string’’. There is no default value. Usually a name is assigned to a window so that the window manager can use a text representation when the window is iconified. This option is unused since you can’t iconify the background.

-display *display*
Specifies the server to connect to; see *X(1)*.

SEE ALSO

X11(7), **xset**(1), **xrdb**(1)

COPYRIGHT

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xsol – play solitaire

SYNOPSIS

xsol [–**display** *connection*] [–**nodrag**] [**[–,+]r**]

DESCRIPTION

xsol plays a solitaire game simliar to Klondike. The playing field is made up of seven slots, where stacks are built in descending value with alternating suits. Aces are built on at the top, and ascending order in the same suit. Kings can be moved to any empty space in the playing field. The deck is gone through only once, card by card.
The cards are moved using the Left mouse button (Button1). Pressing the button selects the card, and it (and any cards on it) can then by dragged to its destination, where releasing will place them. The deck cards are selected by clicking on them.

OPTIONS

–**display** *connection*
Connect to X server display, *connection*.

–**nodrag**
A button press selects the card, and a second press places it.

[–,+]r
Turns reverse video on or off to make cards more readable on monochrome or gray scale devices.

SEE ALSO

X11(7)

COPYRIGHT

NAME

XSolarisGetVisualGamma – obtain gamma information for a visual

SYNTAX

Status
XSolarisGetVisualGamma(Display **display*, int *screen_number*,
Visual **visual*, double **gamma*)

ARGUMENTS

display	Specifies the connection to the X server.
screen_number	Specifies the number of the screen.
visual	Specifies the visual.
gamma	Returns the gamma value for the specified visual.

DESCRIPTION

The **XSolarisGetVisualGamma** function returns the gamma value of a specified visual. This value is the exponent of the power function describing the intensity response of colors displayed using that visual. This is the intensity response of the entire path from the frame buffer pixel store through the monitor. The equation of the intensity response is:

IntensityOut = (*FramebufferColor*)*******gamma*

(i.e. the color in the frame buffer raised to the power of gamma).

FramebufferColor refers to the the RGB values stored in the frame buffer pixel store and processed by any color mapping LUTs that are in the output path.

Note: the term "gamma" used here refers the mapping applied along the entire path, not merely the exponent of the gamma correction function.

The gamma value returned defines the color-to-intensity mapping for all three channels: red, green, and blue.

A status of **Success** is returned if the function was able to determine the gamma successfully. If a request failure occured while calling the function, a **BadAccess** error code is returned. If there is an internal inconsistency (e.g. the gamma value for one of the color channels is different from the others) a **BadMatch** error code is returned. Whenever an error code is returned, the argument *gamma* is left untouched.

The gamma value returned represents the best information available on the intensity response of the visual. Depending on the device, it may or may not include the actual monitor characteristics (some devices have no way of determining the type of monitor so they may assume a default monitor gamma). As such, it represents the system's "best guess" about the intensity response. Since this function gets its information from the same property used by Solaris color managment systems, if more accurate information on the monitor response is configured or calibrated through these systems, this function will return a more accurate value for gamma.

If the intensity mapping is not a power function, the returned gamma value may only be approximate. This should usually happen only when the device gamma correction has been incorrectly configured.

To use this function, an application should link with **libXmu**.

SEE ALSO

Solaris X Window System Developers Guide
(Visuals and Display Devices chapter)

NAME

XSolarisOvlCopyAreaAndPaintType – copies the given area and paint type data from one pair of drawables to another

SYNOPSIS

void XSolarisOvlCopyPaintType (Display **display*, **Drawable** *colorsrc,* **Drawable** *painttypesrc,* **Drawable** *colordst,* **Drawable** *painttypedst,* **GC** *colorgc,* **GC** *painttypegc,* **int** *colorsrc_x,* **int** *colorsrc_y,* **int** *painttypesrc_x,* **int** *painttypesrc_y,* **unsigned** *int width,* **unsigned** *int height,* **int** *colordst_x,* **int** *colordst_y,* **int** *painttypedst_x,* **int** *painttypedst_y,* **unsigned long** *action,* **unsigned long** *plane)*

Arguments

display Specifies the connection to the X server.

colorsrc The color information source drawable.

painttypesrc
The paint type information source drawable.

colordst The color information destination drawable.

painttypedst
The paint type information destination drawable. If *colordst* is an overlay, this drawable will be ignored.

colorgc The GC to use for the color information copy.

painttypegc
The GC to use to fill areas in painttypedst. If *colordst/painttypedst* is an overlay, this GC will be ignored.

colorsrc_x, colorsrc_y
The X and Y coordinates of the upper-left corner of the source rectangle for color information relative to the origin of the color source drawable.

painttypesrc_x, painttypesrc_y
The X and Y coordinates of the upper-left corner of the source rectangle for paint type information relative to the origin of the paint type source drawable.

width, height
The dimensions in pixels of all the source and destination rectangles.

colordst_x, colordst_y
The X and Y coordinates of the upper-left corner of the destination rectangle for color information relative to the origin of the color destination drawable.

painttypedst_x, painttypedst_y
The X and Y coordinates of the upper-left corner of the destination rectangle for paint type information relative to the origin of the paint type destination drawable. If *colordst/painttypedst* is an overlay, *colordst_x* and *colordst_y* will be used.

action Specifies which paint type data is to be copied. This can be one of **XSolarisOvlCopyOpaque, XSolarisOvlCopyTransparent,** or **XSolarisOvlCopyAll.**

plane Specifies the source bit-plane in *painttypesrc* to be used as paint type information when *painttypesrc* is not an overlay.

DESCRIPTION

This routine copies the specified area of *colorsrc* to the specified area of *colordst*. If *colordst* is not an overlay, it also fills the specified areas of *painttypedst* according to the paint type information specified in *painttypesrc*.

colorsrc can be any depth drawable or an overlay window. *painttypesrc* can be any drawable or an overlay window. If *painttypesrc* is not an overlay window, the bit-plane of *painttypesrc* specified in *plane* is treated as if it were paint type data and it is used for the copy. *plane* must have only one bit set in this case. *colordst* can be any drawable, but must be of the same depth and have the same root as *colorsrc,* otherwise **BadMatch** is generated. If *colordst* is an overlay, then *painttypedst* is ignored, otherwise *painttypedst* can be any type of drawable.

The following table summarizes the possible combinations of sources and destinations and their respective actions. The left side of the table shows the possible *colorsrc/painttypesrc* combinations and the top of the table shows the possible *colordst/painttypedst* combinations. The actions, A1-A8, are explained below the table. An Impossible entry in the table indicates that the given combination is impossible since the *painttypedst* is ignored when the *colordst* is an overlay.

	Overlay/ Overlay	**Overlay/ Drawable**	**Drawable/ Overlay**	**Drawable/ Drawable**
overlay / overlay	A1	Impossible	A5	A5
overlay / drawable	A2	Impossible	A6	A6
drawable / overlay	A3	Impossible	A7	A7
drawable / drawable	A4	Impossible	A8	A8

A1—The paint type information from *painttypesrc* is used as a mask to copy the color information from *colorsrc* to *colordst.* Opaque pixels in *painttypesrc* cause the corresponding pixel in *colorsrc* to be copied to *colordst,* transparent pixels cause the corresponding pixel in *colordst* to be made transparent. If a transparent pixel from *colorsrc* is copied to *colordst,* the actual color transferred will be undefined.

A2—Same as A1 except that the paint type information is extracted from the bit-plane of *painttypesrc* specified by *plane.* A bit value of 1 indicates an opaque pixel whereas a bit value of 0 indicates transparent.

A3—Same as A1 except that a non-overlay drawable is used to obtain the color information so there will be no undefined colors due to transparent pixels.

A4—Same as A3 except that the paint type information is taken from the specified bit-plane of *painttypesrc* as in A2.

A5—The paint type information from *painttypesrc* is used as a mask to copy the color information from *colorsrc to colordst* as in A1. In addition, the paint type information controls rendering to the *painttypedst* drawable as in **XSolarisOvlCopyPaintType**(3)**.**

A6—Same as A5 except that the paint type information is taken from the specified bit-plane of *painttypesrc* as in A2.

A7—Same as A5 except that there will be no undefined colors due to transparent color source pixels.

A8—Same as A7 except that the paint type information is taken from the specified bit-plane of *painttypesrc* as in A2.

The *action* argument specifies whether opaque paint (**XSolarisOvlCopyOpaque**), transparent paint (**XSolarisOvlCopyTransparent**), or both (**XSolarisOvlCopyAll**) should be copied. This allows a client to accumulate opaque or transparent paint.

NoExpose and *GraphicsExpose* events are generated in the same manner as **XSolarisOvlCopyPaintType**(3).

If an overlay is used for the *colordst* argument, the *painttypedst, painttypegc, painttypedst_x* and *painttypedst_y* arguments will all be ignored. A NULL pointer can be used for *painttypegc* and a value of None can be used for *painttypedst.* The overlay will have the exact paint type defined by the pixels in the area specified in *painttypesrc.* The color information copy will not affect the destination paint type.

You can use **XSolarisOvlCopyAreaAndPaintType** to combine an image in the client's memory space (consisting of color and / or paint type information) with a rectangle of the specified overlay window. To do this, first move the image and paint type data into the server: use **XPutImage** to copy the data into 2 pixmaps of the appropriate depths. Then call **XSolarisOvlCopyAreaAndPaintType** with the color and paint type drawables to copy information to the overlay.

You can also use **XSolarisOvlCopyAreaAndPaintType** to retrieve pixel information (color and / or paint type information) from a specified drawable. To do this, call **XSolarisOvlCopyAreaAndPaintType** with two separable destination drawables. Then call **XGetImage** on each of the drawables, to get the data from the server into the client's memory space.

This function uses these GC components from colorgc: function, plane-mask, subwindow-mode, graphics-exposures, clip-x-origin, clip-y-origin, and clip-mask. If *colordst* is not an overlay then this function will use these GC components from *painttypegc:* function, plane-mask, fill-style, subwindow-mode, clip-x-origin, clip-y-origin, and clip-mask. In addition, it may also use these GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, and tile-stipple-y-origin.

XSolarisOvlCopyAreaAndPaintType can generate **BadDrawable, BadGC, BadMatch,** and **BadValue** errors.

ERRORS

BadDrawable
BadGC
BadMatch
BadValue

NAME

XSolarisOvlCreateWindow – renders opaque and transparent paint into the destination drawable based on the paint type attributes of the pixels in the source drawable

SYNOPSIS

void XSolarisOvlCopyPaintType (Display **display*, **Drawable** *src*, **Drawable** *dst*, **GC** *gc*, **int** *src_x*, **int** *src_y*, **unsigned** *int width*, **unsigned** *int height*, **int** *dest_x*, **int** *dest_y*, **unsigned long** *action*, **unsigned long** *plane)*

Arguments

display	Specifies the connection to the X server.
src	Specifies the source drawable from which to obtain the paint type information.
dst	Specifies the destination drawable.
gc	Specifies the GC.
src_x, *src_y*	Specify the *x* and *y* coordinates of the upper-left corner of the source rectangle relative to the origin of the source drawable.
width, *height*	Specify the width and height of both the source and destination rectangles.
dest_x, *dest_y*	Specify the *x* and *y* coordinates of the upper-left corner of the destination rectangle relative to the origin of the destination drawable.
action	Specifies which paint type data is to be copied. This can be one of **XSolarisOvlCopyOpaque, XSolarisOvlCopyTransparent,** or **XSolarisOvlCopyAll.**
plane	Specifies the bit-plane of the *src* drawable to be used as paint type information when the source is not an overlay.

DESCRIPTION

This routine uses the paint type information of the specified rectangle of *src* to control a fill operation in the specified rectangle of *dst*. *src* and *dst* can be any type of drawable. If *src is an overlay, the paint* source of the copy, and the color information is ignored. If *src* is any other type of drawable, the bit-plane specified in *plane* is treated as if it were paint type data and it is used for the copy. *plane* must have only one bit set in this case. The following table summarizes the possible combinations of *src* and *dst* and their actions. The left side of the table shows the possible *src* combinations. The top of the table shows the possible *dst* combinations. The actions, A1-A4, are explained below the table.

Source/Destination	**Overlay**	**Drawable**
overlay	A1	A2
drawable	A3	A4

A1—Opaque pixels in the source overlay cause the corresponding pixels in the destination to be filled with opaque color as specified by the fill attributes of the GC. Transparent pixels in the source cause the corresponding pixels in the destination to be filled with transparent paint.

A2—Opaque pixels in the source overlay cause the corresponding pixels in the destination to be filled according to the fill attributes of the GC. Transparent pixels in the source overlay cause the corresponding pixels in the destination to be filled according to the same fill attributes of the GC, but with the foreground and background pixels swapped.

A3—The pixels in the destination overlay are filled with opaque paint or made transparent as in A1 above depending on the bit values of the source drawable's plane. Bit values of 1 in the source are treated as if they were opaque pixels and bit values of 0 are treated as if they were transparent.

A4—The pixels in the destination drawable are filled with paint as in A2 above depending on the bit values of the source drawable's plane. Bit values of 1 in the source bit plane are treated as if they were opaque pixels and bit values of 0 are treated as if they were transparent.

The *action* argument specifies whether opaque paint (**XSolarisOvlCopyOpaque**), transparent paint (**XSolarisOvlCopyTransparent**), or both (**XSolarisOvlCopyAll**) should be operated upon. This allows a client to accumulate opaque or transparent paint.

src and *dst* must have the same screen, or a **BadMatch** error results.

If portions of the source rectangle are obscured or are outside the boundaries of the source drawable, the server generates exposure events, using the same semantics as **XCopyArea.**

This routine uses these GC components: function, plane-mask, fill-style, subwindow-mode, graphics-exposures, clip-x-origin, clip-y-origin, and clip-mask. It might use these GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin.

XSolarisOvlCopyPaintType can generate **BadDrawable, BadGC, BadMatch,** and **BadValue** errors.

ERRORS

BadDrawable
BadGC
BadMatch
BadValue

NAME

XSolarisOvlCreateWindow – creates an overlay window

SYNOPSIS

Window XSolarisOvlCreateWindow (Display **display,* **Window** *parent,* **int** *x,* **int** *y,* **unsigned** *int width,* **unsigned** *int height,* **unsigned** *int border_width,* **int** *depth,* **unsigned** *int class,* **Visual** **visual,* **unsigned long** *valuemask,* **XSetWindowAttributes** **attr)*

Arguments

The arguments for this routine are exactly the same as XCreateWindow.

display Specifies the connection to the X server.

parent Specifies the parent window.

x, *y* Specifies the coordinates of the upper-left pixel of this window, relative to the parent window.

width, *height*
Specifies the width and height, in pixels, of the window.

border_width
Specifies the width, in pixels, of the window's borders.

depth Specifies the depth of the window.

class Specifies the class of the window. If it is not **InputOutput,** the window will not be an overlay window.

visual Specifies a pointer to the visual structure for this window.

valuemask
Specifies which window attributes are defined in the attr argument.

attr Specifies the attributes of the window.

DESCRIPTION

This routine creates an overlay window with the given characteristics. It behaves exactly as its counterpart **XCreateWindow,** except the newly created window can be rendered into with both opaque and transparent paint, and the background is transparent.

SEE ALSO

XCreateWindow(3)

NAME

XSolarisOvlGetPaintType – gets the current paint type set in the GC

SYNOPSIS

XSolarisOvlPaintType
  XSolarisOvlGetPaintType (Display **display*, **GC** *gc)*

Arguments

display	Specifies the connection to the X server.
gc	The GC to be inquired from.

DESCRIPTION

This routine returns the current element of type **XSolarisOvlPaintType** associated with the given *gc*.

XSolarisOvlPaintType defines the paint type in each GC. An enumeration defining two types of selections that can be done in **XSolarisOvlPaintType**

```
typedef enum {
        XSolarisOvlPaintTransparent,
        XSolarisOvlPaintOpaque,
} XSolarisOvlPaintType;
```

NAME

XSolarisOvlIsOverlayWindow – indicates whether a given window is an overlay window

SYNOPSIS

Bool XSolarisOvlIsOverlayWindow (Display **display*, **Window** *w)*

Arguments

display	Specifies the connection to the X server.
w	Specifies the window.

DESCRIPTION

Returns **True** if the given window *w* is an overlay window. Otherwise returns **False.**

NAME

XSolarisOvlSelectPair – selects an optimal overlay / underlay visual pair that best meets the criteria

SYNOPSIS

XSolarisOvlSelectStatus
XSolarisOvlSelectPair (Display *∗display,* **int** *screen,* **int** *numCriteria,* **XSolarisOvlPairCriteria** *∗pCriteria,* **XVisualInfo** *∗ovVisinfoReturn,* **XVisualInfo** *∗unVisinfoReturn,* **unsigned long** *∗unmetOvCriteriaReturn,* **unsigned long** *∗unmetUnCriteriaReturn)*

Arguments

display Specifies the connection to the X server.

screen An integer specifying the screen on which the visuals are to be searched.

numCriteria
The number of **XSolarisOvlPairCriteria** structures in the *pCriteria* array.

pCriteria
An array of pair criteria structures in priority order from high to low specifying the criteria to be used in selecting the pair.

ovVisinfoReturn
A pointer to a caller provided *XVisualInfo* structure. On successful return, this structure contains a description of the chosen overlay visual.

unVisinfoReturn
A pointer to a caller provided *XVisualInfo* structure. On successful return, this structure contains a description of the chosen underlay visual.

unmetOvCriteriaReturn
A pointer to a bitmask that describes the criteria that were not satisfied for the overlay visual. This return argument is only meaningful when the routine returns a value of **XSolarisOvlQualifiedSuccess,** or **XSolarisOvlCriteriaFailure.**

unmetUnCriteriaReturn
A pointer to a bitmask that describes the criteria that were not satisfied for the underlay visual. This return argument is only meaningful when the routine returns a value of **XSolarisOvlQualifiedSuccess,** or **XSolarisOvlCriteriaFailure.**

Argument Types

See the Description section for a full description of how these types should be used.

XSolarisOvlPairCriteria
A structure defining various criteria to be used during visual selection, along with indications of the stringency of the criteria.

```
typedef struct {
        XSolarisOvlVisualCriteriaoverlayCriteria;
        XSolarisOvlVisualCriteriaunderlayCriteria;
} XSolarisOvlPairCriteria;
```

XSolarisOvlVisualCriteria is defined in the specification of **XSolarisOvlSelectPartner**(3)**.**

Return Types

XSolarisOvlSelectStatus
Refer to the specification of **XSolarisOvlSelectPartner**(3) for the definition of this type.

XSolarisOvlSuccess is returned if the search is completely successful in finding a pair that meets all hard and soft criteria of one of the **XSolarisOvlPairCriteria** structures.

XSolarisOvlQualifiedSuccess is returned if the chosen pair satisfies all hard criteria of one of the **XSolarisOvlPairCriteria** structures, but doesn't meet all soft criteria. In this case, *unmetOvCriteriaReturn* and *unmetUnCriteriaReturn* contains the logical OR of the soft criteria that were not met for the overlay and underlay, respectively.

XSolarisOvlCriteriaFailure indicates that no pair could be found that meets all the hard criteria of any of the **XSolarisOvlPairCriteria** structures. In this case, *unmetOvCriteriaReturn* and *unmetUnCriteriaReturn* contains the logical OR of the hard criteria that were not met by the **XSolarisOvlPairCriteria** structure with the fewest hard failures, for the overlay and underlay, respectively.

XSolarisOvlFailure is returned if some other error is encountered besides criteria match failure.

DESCRIPTION

This routine is similar to **XSolarisOvlSelectPartner**(3)**.** However, instead of selecting a partner visual given another visual, this routine simultaneously selects both the overlay and underlay visual from the set of all visual pairs for the given screen. The pair selected will be the one that best matches the given criteria.

The client is assured that, short of X errors not related to overlays, it can successfully create windows with the returned visuals.

This routine searches through all optimal visual pairs for a given screen, and then through all pairs of visuals (optimal and non-optimal), applying the specified criteria. These criteria are specified in *pCriteria.* Each element of *pCriteria* specifies criteria for both the overlay and underlay. It returns a success or failure status depending on whether it finds a pair that meets all the given criteria.

The selected pair will have an overlay that satisfies all the hard criteria specified for the overlay. The pair will have an underlay visual that satisfies all the hard criteria for the underlay. The attributes of the overlay visual are returned in *ovVisinfoReturn.* Likewise, the attributes of the underlay visual are specified in *unVisinfoReturn.* If two or more pairs are found that meet all of the hard criteria (both overlay and underlay) and the same number of soft criteria (either overlay or underlay), one of them will be chosen and returned. It is implementation dependent which one is chosen.

Like **XSolarisOvlSelectPartner**(3)**, XSolarisOvlSelectPair** supports a degradation sequence of criteria sets. This means that multiple criteria sets can be specified in a single call. First, an attempt is made to find a pair matching the first criteria set for both the overlay and the underlay. If a pair is found which meets all of the hard criteria of the first set, this pair is chosen. If no pair meets all hard criteria of the first set, a search is performed using the second criteria set. This process continues until either a pair is found that meets the all of the hard criteria of some criteria set, or all sets have been used to search. This degradation sequence allows clients to specify the criteria for the most preferred pair as the first criteria set. Pairs that are acceptable but which are less desirable can be specified in criteria sets following the first. This allows the search to proceed through a progressive relaxation in the client's requirements for the pair with a single subroutine call.

The criteria masks that can be specified are described in the specification of **XSolarisOvlSelectPartner**(3)**.**

NAME

XSolarisOvlSelectPartner – returns the overlay / underlay visual that best meets the criteria

SYNOPSIS

XSolarisOvlSelectStatus
  XSolarisOvlSelectPartner (Display ∗*display,* **int** *screen,* **VisualID** *vid,* **XSolarisOvlSelectType** *seltype,* **int** *numCriteria,* **XSolarisOvlVisualCriteria** ∗*pCriteria,* **XVisualInfo** ∗*visinfoReturn,* **unsigned long** ∗*unmetCriteriaReturn)*

Arguments

display Specifies the connection to the X server.

screen An integer specifying the screen for the visual *vid.*

vid The XID of the visual to find a partner for.

seltype The type of selection that is to be done.

numCriteria
  The number of **XSolarisOvlVisualCriteria** structures in the *pCriteria array*.

pCriteria
  An array of criteria structures in priority order from high to low specifying the criteria to be used in selecting the visual.

visinfoReturn
  A pointer to a caller provided **XVisualInfo** structure. On successful return, this structure contains a description of the chosen visual.

unmetCriteriaReturn
  A pointer to a bitmask that describes the criteria that were not satisfied. This return argument is only meaningful when the routine returns a value of **XSolarisOvlQualifiedSuccess,** or **XSolarisOvlCriteriaFailure.**

Argument Types

See the **XSolarisOvlSelectPartner** Description section for a full description of how these types should be used.

XSolarisOvlSelectType

An enumeration defining two types of selections that can be done in **XSolarisOvlSelectPartner.**

```
typedef enum {
        XSolarisOvlSelectBestOverlay,
        XSolarisOvlSelectBestUnderlay,
} XSolarisOvlSelectType;
```

XSolarisOvlVisualCriteria

A structure defining various criteria to be used during visual selection, along with indications of the stringency of the criteria.

```
typedef struct {
        unsigned long   hardCriteriaMask;
        unsigned long   softCriteriaMask
        int             c_class;
        unsigned int    depth;
        unsigned int    minColors;
        unsigned int    minRed;
        unsigned int    minGreen;
        unsigned int    minBlue;
        unsigned int    minBitsPerRGB;
        unsigned int    minBuffers;
} XSolarisOvlVisualCriteria;
```

hardCriteriaMask and *softCriteriaMask* are bitmasks whose values can be the logical OR of any of the following bitmasks:

```
#define XSolarisOvlVisualClass          (1L<<0)
#define XSolarisOvlDepth                (1L<<1)
#define XSolarisOvlMinColors            (1L<<2)
#define XSolarisOvlMinRed               (1L<<3)
#define XSolarisOvlMinGreen             (1L<<4)
#define XSolarisOvlMinBlue              (1L<<5)
#define XSolarisOvlMinBitsPerRGB        (1L<<6)
#define XSolarisOvlMinBuffers           (1L<<7)
#define XSolarisOvlUnsharedPixels       (1L<<8)
#define XSolarisOvlUnsharedColors       (1L<<9)
#define XSolarisOvlPreferredPartner     (1L<<10)
```

These are described in the **XSolarisOvlSelectPartner** Description documentation that follows.

Return Types

XSolarisOvlSelectStatus

A value that indicates whether the routine succeeded in finding a visual and, if it failed, the reason for the failure. The return value can be one of:

```
typedef enum {
        XSolarisOvlSuccess,
        XSolarisOvlQualifiedSuccess,
        XSolarisOvlCriteriaFailure,
        XSolarisOvlFailure,
} XSolarisOvlSelectStatus;
```

XSolarisOvlSuccess is returned if the search is completely successful in finding a visual that meets all hard and soft criteria of one of the **XSolarisOvlVisualCriteria** structure.

XSolarisOvlQualifiedSuccess is returned if the chosen visual satisfies all hard criteria of one of the **XSolarisOvlVisualCriteria** structure, but doesn't meet all soft criteria. In this case, *unmetCriteriaReturn* contains the logical OR of the soft criteria that were not met.

XSolarisOvlCriteriaFailure indicates that no visual could be found that meets all the hard criteria of any of the **XSolarisOvlVisualCriteria** structures. In this case, *unmetCriteriaReturn* contains the logical OR of the hard criteria that were not met for the **XSolarisOvlVisualCriteria** structure with the fewest hard criteria not met.

XSolarisOvlFailure is returned if some other error is encountered besides criteria match failure.

DESCRIPTION

Portable applications using overlays may wish to search for an appropriate overlay visual to use for a given underlay visual, or vice-versa. Each X screen supporting the overlay extension defines a set of overlay visuals whose windows are best for use as children of underlay windows. For each underlay visual, there is a set of *optimal* overlay visuals. Together, all combinations of underlay visuals and their optimal overlay visuals form the set of optimal overlay / underlay pairs for that screen. The overlay and underlay visuals of an optimal pair are said to be *partners* of each other.

XSolarisOvlSelectPartner allows the client to select, given an underlay visual, an *optimal* overlay that meets certain criteria. Inversely, it also allows the client to select an optimal underlay visual given an overlay visual.

The client is assured that, short of X errors not related to overlays, it can successfully create a window with the returned visual.

This routine searches through the optimal partners of the given visual, applying the criteria specified in *pCriteria.* It returns a success or failure status depending on whether it finds a visual that meets the criteria.

A criterion can be one of two types:

1. Hard Criterion

   A criterion that must be satisfied. Only visuals that meet hard criteria are candidates for successful matches.

2. Soft Criterion

   A desirable criterion, but one which is not required. The visual that matches all hard criteria and the most soft criteria is chosen. Its attributes are returned in *visinfoReturn.* If two or more visuals are found that meet all of the hard criteria and the same number of soft criteria, one of them will be chosen and returned. It is implementation dependent which one is chosen.

XSolarisOvlSelectPartner supports a degradation sequence of criteria sets. This means that multiple criteria sets can be specified in a single call. First, an attempt is made to find a visual matching the first criteria set. If a visual is found which meets all of the hard criteria of the first set, this visual is chosen. If no visual met all hard criteria of the first set, a search is performed using the second criteria set. This process continues until either a visual is found that meets the hard criteria of some criteria set, or all sets have been used to search. This degradation sequence allows clients to specify the criteria for the most preferred visual as the first criteria set. Visuals that are acceptable but which are less desirable can be specified in criteria sets following the first. This allows the search to proceed through a progressive relaxation in the client's requirements for the visual with a single subroutine call.

Any of the possible criteria can be specified either as a hard or soft criteria for a particular criteria set. For a given set, *hardCriteriaMask* is the logical OR of the criteria bitmasks that are to be applied as hard criteria during the search. Likewise, *softCriteriaMask* is the logical OR of the soft criteria bitmasks.

Some criteria have values associated with them. These values are provided by other data members in the **XSolarisOvlVisualCriteria** structure. In the criteria descriptions which follow, these data members are mentioned where applicable.

XSolarisOvlVisualClass specifies that the client desires the selected visual to have a specific visual class. The required class is specified in *c_class.*

The following criteria interact within one another: **XSolarisOvlDepth, XSolarisOvlMinColors, XSolarisOvlMinRed, XSolarisOvlMinGreen,** and **XSolarisOvlMinBlue.** Typically, only some subset of these should be specified. **XSolarisOvlDepth** specifies that the depth of the selected visual is to be equal to depth. **XSolarisOvlMinColors** specifies that the selected visual is to have at least minColors number of total displayable colors. **XSolarisOvlMinRed, XSolarisOvlMinGreen,** and **XSolarisOvlMinBlue** can be used to indicate more specific color requirements for *DirectColor* or *TrueColor* visuals. Their corresponding values are specified in *minRed, minGreen,* and *minBlue,* respectively. These indicate that the selected visual must have at least the specified number of reds, greens, and/or blues.

XSolarisOvlMinBitsPerRGB specifies that the selected visual is to have at least *minBitsPerRGB* of color channel output from colormaps created on that visual.

XSolarisOvlMinBuffers specifies that the client desires the selected visual to be able to be assigned at least *minBuffers* number of accelerated MBX image buffers.

XSolarisOvlUnsharedPixels selects partner visuals whose window pixels don't lie in the same drawing plane groups as the window pixels of the argument visual *vid.* If a visual uses the same drawing plane group as the argument visual it is not matched by this criterion.

XSolarisOvlUnsharedColors selects partner visuals whose window pixel colors can be displayed simultaneously when the overlay/underlay window pair has the colormap focus. If a visual shares the same color LUT pool and that pool has only one color LUT in it as the argument visual it is not matched by this criterion.

If either *hardCriteriaMask* of a criteria set is to 0, any visual will match that criteria set with a hard match. Likewise, setting the *softCriteriaMask* of a criteria set to 0, is sufficient to guarantee at least a soft match for that criteria set.

NAME

XSolarisOvlSetPaintType – specifies the type of paint rendered by subsequent Xlib drawing with the given GC

SYNOPSIS

void XSolarisOvlSetPaintType (Display **display*, **GC** *gc*, **XSolarisOvlPaintType** *paintType)*

Arguments

display	Specifies the connection to the X server.
gc	Specifies the affected GC.
paintType	Specifies the type of paint rendered by subsequent Xlib drawing routines using the specified GC.

DESCRIPTION

This routine controls the type of paint rendered by an Xlib GC. It controls whether Xlib drawing routines using this GC produce pixels on overlay windows that are opaque or transparent. The paint type specified applies to the GC until it is changed by another call to this routine. The paint type attribute applies to both the foreground and background GC attributes. If the value of *paintType* is **XSolarisOvlPaintOpaque,** the pixels generated by subsequent Xlib drawing routines with this GC will be opaque. This means the pixels will obscure underlying pixels. If the value of *paintType* is **XSolarisOvlPaintTransparent,** the pixels generated by subsequent Xlib drawing routines with this GC will be transparent. This means that, for these pixels, the color of the underlying pixels will be displayed. By default, a GC renders opaque paint.

NAME

XSolarisOvlSetWindowTransparent – sets the background state of an overlay window to be transparent

SYNOPSIS

void XSolarisOvlSetWindowTransparent (Display **display*, **Window** *w)*

Arguments

display	Specifies the connection to the X server.
w	Specifies the window.

DESCRIPTION

This routine sets the background state of the given overlay to be transparent. Any background rendering that occurs after this request will cause the background to be transparent. Use **XChangeWindowAttributes**(3)**, XSetWindowBackground**(3)**,** or **XSetWindowBackgroundPixmap**(3) to change background state to any other value.

If *w* is not an overlay window, **BadMatch** is generated.

ERRORS

BadMatch

NAME

xstdcmap – X standard colormap utility

SYNOPSIS

xstdcmap [-all] [-best] [-blue] [-default] [-delete *map*] [-display *display*] [-gray] [-green] [-help] [-red] [-verbose]

DESCRIPTION

The **xstdcmap** utility can be used to selectively define standard colormap properties. It is intended to be run from a user's X startup script to create standard colormap definitions in order to facilitate sharing of scarce colormap resources among clients. Where at all possible, colormaps are created with read-only allocations.

OPTIONS

The following options may be used with **xstdcmap:**

–all This option indicates that all six standard colormap properties should be defined on each screen of the display. Not all screens will support visuals under which all six standard colormap properties are meaningful. **xstdcmap** will determine the best allocations and visuals for the colormap properties of a screen. Any previously existing standard colormap properties will be replaced.

–best This option indicates that the RGB_BEST_MAP should be defined.

–blue This option indicates that the RGB_BLUE_MAP should be defined.

–default
This option indicates that the RGB_DEFAULT_MAP should be defined.

–delete *map*
This option specifies that a standard colormap property should be removed. *map* may be one of: default, best, red, green, blue, or gray.

–display *display*
This option specifies the host and display to use; see **X11**(7).

–gray This option indicates that the RGB_GRAY_MAP should be defined.

–green This option indicates that the RGB_GREEN_MAP should be defined.

–help This option indicates that a brief description of the command line arguments should be printed on the standard error. This will be done whenever an unhandled argument is given to *xstdcmap.*

–red This option indicates that the RGB_RED_MAP should be defined.

–verbose
This option indicates that **xstdcmap** should print logging information as it parses its input and defines the standard colormap properties.

ENVIRONMENT

DISPLAY
to get default host and display number.

SEE ALSO

X11(7)

COPYRIGHT

AUTHOR

Donna Converse, MIT X Consortium

NAME

Xsun – Solaris server for X Version 11

SYNOPSIS

Xsun [option] ...

DESCRIPTION

Xsun is the Solaris server for Version 11 of the X window system on Solaris hardware. It is normally started by **xinit**(1) via **openwin**(1).

NETWORK CONNECTIONS

In addition to the network connections described in **Xserver**(1), **Xsun** provides the following connection:

Shared Memory
: Sun provides a shared memory transport mechanism via the SUN_SME extension. This extension provides the capability of sending client requests to the server via shared memory. Shared memory is used for client requests only. Replies from the server and events are sent via the default transport mechanism. To enable this transport mechanism, one has to set the DISPLAY environment variable to :x.y (where x is the display number and y is the screen number), and the environment variable XSUNTRANSPORT to "shmem". The size of the segment can be set by setting enviroment variable XSUNSMESIZE. By default, it is set to "64" which implies that the size of the shared memory segment is 64K.

OPTIONS

In addition to the options described in **Xserver**(1), **Xsun** accepts the following command-line switches:

–**dev** *filename*
: This option specifies the name of the framebuffer device file to be used instead of the default framebuffer /dev/fb. Multiple instances of this option indicate multiple screens on the same server. After each **-dev** option a list of modifiers changes the behavior of the named device.

[*left* | *right* | *top* | *bottom*]
: Specify the position of a given screen in relation to the previous one on the command line. The default *right.*

dpix n The dpi in the x direction for this screen is *n*. The default is 90.

dpiy n The dpi in the y direction for this screen is *n*. The default is 90.

defclass [*GrayScale* | *StaticGray* | *PseudoColor* | *StaticColor* | *DirectColor* | *TrueColor*] Use the specified visual as the default visual. The default is device dependent.

defdepth n
: A visual of depth n is the default visual. The default is device dependent.

grayvis Only report GrayScale and/or StaticGray visuals.

The following is an example of the -dev option that might be used on a system with a cg6 and a bw2:

```
-dev /dev/cgsix0 defclass GrayScale -dev /dev/bwtwo0 right
```

The **Xsun** server also supports the format used by the X11R5 sample X Server. Multiple screen systems are specified by using the following syntax on the command line:

```
-dev <device 1>:<device 2>:...:<device n>
```

The server uses device 1 as screen 0, device 2 as screen 1, etc., and the server assumes that screens are ordered left to right in ascending screen number. This means that the cursor moves off the right side of screen n and onto the left side of screen n + 1. When this format is used, no other **-dev** options are valid.

– **accessX**
This option enables activation of the slowkey and stickykey functionality of the *AccessX* extension using the shift key on the keyboard.

–**ar1** *milliseconds*
Specify amount of time in milliseconds before a pressed key begins to autorepeating. The default is 500 milliseconds. This option is not available on Solaris x86 or PowerPC.

–**ar2** *milliseconds*
specify the interval in milliseconds between autorepeats of pressed keys. The default is 50 milliseconds. This option is not available on Solaris x86 or PowerPC.

–**banner**
Display the OpenWindows banner screen at startup. The banner is displayed by default.

–**dur** *milliseconds*
Set the duration of the bell in units of milliseconds. Default is 100.

–**dpsfileops**
Allow the Display PostScript file operators access to the UNIX file system.

–**flipPixels**
Reverse black and white pixel locations in the colormap. This is not the same as a reverse video option.

–**pit** *percentage*
Set the percentage of the maximum pitch available to the hardware. Sun hardware does not support alternative pitch values.

–**mden** *denominator*
Set the pointer acceleration denominator. The acceleration numerator is set with the -a option described in **Xserver**(1). This permits fractional acceleration such as 3/2 or 1.5. Default value is 1.

–**nobanner**

Do not display the OpenWindows banner screen at startup.

–nominexp
: This option is used to disable "minimized exposure", which is used only by multi-planegroup devices such as gt, cg12 and cg8. "Minimized Exposure" means that the server will not send expose events to windows in one planegroup that are exposed by windows in another planegroup. By default the minimized exposure feature is turned ON.

–sharedretainedpath *directory_path*
: This option is currently supported only for Sun internal software APIs.

SEE ALSO

openwin(1), **X11**(7), **Xserver**(1), **xdm**(1), **xinit**(1)

NAME

xterm – terminal emulator for X

SYNOPSIS

xterm [–*toolkitoption* ...] [–option ...]

DESCRIPTION

The **xterm** program is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 compatible terminals for programs that can't use the window system directly. If the underlying operating system supports terminal resizing capabilities (for example, the SIGWINCH signal in systems derived from 4.3bsd), **xterm** will use the facilities to notify programs running in the window whenever it is resized.

The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height / width), Tektronix graphics will be restricted to the largest box with a 4014's aspect ratio that will fit in the window. This box is located in the upper left area of the window.

Although both windows may be displayed at the same time, one of them is considered the ''active'' window for receiving keyboard input and terminal output. This is the window that contains the text cursor. The active window can be chosen through escape sequences, the ''VT Options'' menu in the VT102 window, and the ''Tek Options'' menu in the 4014 window.

EMULATIONS

The VT102 emulation is fairly complete, but does not support the blinking character attribute nor the double-wide and double-size character sets. **Terminfo** entries that work with **xterm** include ''xterm,'' ''vt102,'' ''vt100'' and ''ansi,'' and **xterm** automatically searches the **/usr/share/lib/terminfo** directory in this order for these entries and then sets the ''TERM'' environment variable (see **terminfo**(4)).

Many of the special **xterm** features (like logging) may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See the "*Xterm Control Sequences*" document.)

The Tektronix 4014 emulation is also fairly good. Four different font sizes and five different lines types are supported. The Tektronix text and graphics commands are recorded internally by **xterm** and may be written to a file by sending the COPY escape sequence (or through the **Tektronix** menu; see below). The name of the file will be ''**COPY***yy*–*MM*–*dd*.*hh*:*mm*:*ss*'', where *yy*, *MM*, *dd*, *hh*, *mm* and *ss* are the year, month, day, hour, minute and second when the COPY was performed (the file is created in the directory **xterm** is started in, or the home directory for a login **xterm**).

OTHER FEATURES

Xterm automatically highlights the text cursor when the pointer enters the window (selected) and unhighlights it when the pointer leaves the window (unselected). If the window is the focus window, then the text cursor is highlighted no matter where the pointer is.

In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which is the same size as the display area of the window. When activated, the current screen is saved and replaced with the alternate screen. Saving of lines scrolled off the top of the window is disabled until the normal screen is restored. The **terminfo**(4) entry for **xterm** allows the visual editor **vi**(1) to switch to the alternate screen for editing and to restore the screen on exit.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows and to specify a new log file name. See *Xterm Control Sequences* for details. Enabling the escape sequence to change the log file name is a compile-time option; by default this escape sequence is ignored for security reasons.

OPTIONS

The **xterm** terminal emulator accepts all of the standard X Toolkit command line options as well as the following (if the option begins with a '**+**' instead of a '–', the option is restored to its default value):

–help This causes **xterm** to print out a verbose message describing its options.

–132 Normally, the VT102 DECCOLM escape sequence that switches between 80 and 132 column mode is ignored. This option causes the DECCOLM escape sequence to be recognized, and the **xterm** window will resize appropriately.

–ah This option indicates that **xterm** should always highlight the text cursor. By default, *xterm* will display a hollow text cursor whenever the focus is lost or the pointer leaves the window.

+ah This option indicates that **xterm** should do text cursor highlighting based on focus.

–b *number*
This option specifies the size of the inner border (the distance between the outer edge of the characters and the window border) in pixels. The default is 2.

–cc *characterclassrange*:*value*[,...]
This sets classes indicated by the given ranges for using in selecting by words. See the section specifying character classes.

–cn This option indicates that newlines should not be cut in line-mode selections.

+cn This option indicates that newlines should be cut in line-mode selections.

–cr *color* This option specifies the color to use for text cursor. The default is to use the same foreground color that is used for text.

–cu This option indicates that **xterm** should work around a bug in the **curses**(3X) cursor motion package that causes the **more**(1) program to display lines that are exactly the width of the window and are followed by a line beginning with a tab to be displayed incorrectly (the leading tabs are not displayed).

+cu This option indicates that that **xterm** should not work around the **curses**(3X) bug mentioned above.

–e *program [arguments ...]*
This option specifies the program (and its command line arguments) to be run in the **xterm** window. It also sets the window title and icon name to be the basename of the program being executed if neither *–T* nor *–n* are given on the command line. **This must be the last option on the command line.**

–fb *font* This option specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font will be produced by overstriking this font. The default is to do overstriking of the normal font.

–j This option indicates that **xterm** should do jump scrolling. Normally, text is scrolled one line at a time; this option allows **xterm** to move multiple lines at a time so that it doesn't fall as far behind. Its use is strongly recommended since it make **xterm** much faster when scanning through large amounts of text. The VT100 escape sequences for enabling and disabling smooth scroll as well as the ''VT Options'' menu can be used to turn this feature on or off.

+j This option indicates that **xterm** should not do jump scrolling.

–l This option indicates that B xterm should send all terminal output to a log file as well as to the screen. This option can be enabled or disabled using the ''VT Options'' menu.

+l This option indicates that **xterm** should not do logging.

–lf *filename*
This option specifies the name of the file to which the output log described above is written. If *file* begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The ability to log to a pipe is a compile-time option which is disabled by default for security reasons. The default filename is ''**XtermLog.***XXXXX*'' (where *XXXXX* is the process id of **xterm**) and is created in the directory from which **xterm** was started (or the user's home directory in the case of a login window).

–ls This option indicates that the shell that is started in the **xterm** window be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the user's .login or .profile).

+ls This option indicates that the shell that is started should not be a login shell (i.e. it will be a normal ''subshell'').

–mb This option indicates that **xterm** should ring a margin bell when the user types near the right end of a line. This option can be turned on and off from the ''VT Options'' menu.

+mb This option indicates that margin bell should not be rung.

–mc *milliseconds*
This option specifies the maximum time between multi-click selections.

–ms *color*
This option specifies the color to be used for the pointer cursor. The default is to use the foreground color.

–nb *number*
This option specifies the number of characters from the right end of a line at which the margin bell, if enabled, will ring. The default is 10.

–rw This option indicates that reverse-wraparound should be allowed. This allows the cursor to back up from the leftmost column of one line to the rightmost column of the previous line. This is very useful for editing long shell command lines and is encouraged. This option can be turned on and off from the ''VT Options'' menu.

+rw This option indicates that reverse-wraparound should not be allowed.

–aw This option indicates that auto-wraparound should be allowed. This allows the cursor to automatically wrap to the beginning of the next line when when it is at the rightmost position of a line and text is output.

+aw This option indicates that auto-wraparound should not be allowed.

–s This option indicates that **xterm** may scroll asynchronously, meaning that the screen does not have to be kept completely up to date while scrolling. This allows **xterm** to run faster when network latencies are very high and is typically useful when running across a very large internet or many gateways.

+s This option indicates that **xterm** should scroll synchronously.

–sb This option indicates that some number of lines that are scrolled off the top of the window should be saved and that a scrollbar should be displayed so that those lines can be viewed. This option may be turned on and off from the ''VT Options'' menu.

+sb This option indicates that a scrollbar should not be displayed.

–sf This option indicates that Sun Function Key escape codes should be generated for function keys.

+sf This option indicates that the standard escape codes should be generated for function keys.

–si This option indicates that output to a window should not automatically reposition the screen to the bottom of the scrolling region. This option can be turned on and off from the ''VT Options'' menu.

+si This option indicates that output to a window should cause it to scroll to the bottom.

–sk This option indicates that pressing a key while using the scrollbar to review previous lines of text should cause the window to be repositioned automatically in the normal position at the bottom of the scroll region.

+sk This option indicates that pressing a key while using the scrollbar should not cause the window to be repositioned.

–sl *number*
This option specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.

–t This option indicates that **xterm** should start in Tektronix mode, rather than in VT102 mode. Switching between the two windows is done using the ''Options'' menus.

+t This option indicates that **xterm** should start in VT102 mode.

–tm *string*
: This option specifies a series of terminal setting keywords followed by the characters that should be bound to those functions, similar to the *stty* program. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ˆchar (e.g. ˆc or ˆu) and ˆ? may be used to indicate delete.

–tn *name*
: This option specifies the name of the terminal type to be set in the TERM environment variable. This terminal type must exist in the **termcap**(5) database and should have *li#* and *co#* entries.

–ut This option indicates that **xterm** shouldn't write a record into the the system log file */etc/utmp*.

+ut This option indicates that **xterm** should write a record into the system log file */etc/utmp*.

–vb This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed.

+vb This option indicates that a visual bell should not be used.

–wf This option indicates that **xterm** should wait for the window to be mapped the first time before starting the subprocess so that the initial terminal size settings and environment variables are correct. It is the application's responsibility to catch subsequent terminal size changes.

+wf This option indicates that **xterm** show not wait before starting the subprocess.

–C This option indicates that this window should receive console output. This is not supported on all systems. To obtain console output, you must be the owner of the console device, and you must have read and write permission for it. If you are running X under *xdm* on the console screen you may need to have the session startup and reset programs explicitly change the ownership of the console device in order to get this option to work.

–S*ccn* This option specifies the last two letters of the name of a pseudoterminal to use in slave mode, plus the number of the inherited file descriptor. The option is parsed ''%c%c%d''. This allows **xterm** to be used as an input and output channel for an existing program and is sometimes used in specialized applications.

The following command line arguments are provided for compatibility with older versions. They may not be supported in the next release as the X Toolkit provides standard options that accomplish the same task.

%*geom* This option specifies the preferred size and position of the Tektronix window. It is shorthand for specifying the ''**tekGeometry*'' resource.

#*geom* This option specifies the preferred position of the icon window. It is shorthand for specifying the ''**iconGeometry*'' resource.

–T *string*

This option specifies the title for **xterm** It is equivalent to –**title**.

–**n** *string*
This option specifies the icon name for **xterm** It is shorthand for specifying the ‘‘**iconName*’’ resource. Note that this is not the same as the toolkit option –**name** (see below). The default icon name is the application name.

–**r** This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to –**reversevideo** or –**rv**.

–**w** *number*
This option specifies the width in pixels of the border surrounding the window. It is equivalent to –**borderwidth** or –**bw**.

The following standard X Toolkit command line arguments are commonly used with **xterm:**

–**bg** *color*
This option specifies the color to use for the background of the window. The default is ‘‘white.’’

–**bd** *color*
This option specifies the color to use for the border of the window. The default is ‘‘black.’’

–**bw** *number*
This option specifies the width in pixels of the border surrounding the window.

–**fg** *color*
This option specifies the color to use for displaying text. The default is ‘‘black.’’

–**fn** *font* This option specifies the font to be used for displaying normal text. The default is *fixed*.

–**name** *name*
This option specifies the application name under which resources are to be obtained, rather than the default executable file name. *Name* should not contain ‘‘.’’ or ‘‘*’’ characters.

–**title** *string*
This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the –**e** option, if any, otherwise the application name.

–**rv** This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–**geometry** *geometry*
This option specifies the preferred size and position of the VT102 window; see **X**(7)

–**display** *display*
This option specifies the X server to contact; see **X**(7)

–**xrm** *resourcestring*
This option specifies a resource string to be used. This is especially useful for

setting resources that do not have separate command line options.

–iconic This option indicates that **xterm** should ask the window manager to start it as an icon rather than as the normal window.

RESOURCES

The program understands all of the core X Toolkit resource names and classes as well as:

iconGeometry (class **IconGeometry)**
Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

termName (class **TermName)**
Specifies the terminal type name to be set in the TERM environment variable.

title (class **Title)**
Specifies a string that may be used by the window manager when displaying this application.

ttyModes (class **TtyModes)**
Specifies a string containing terminal setting keywords and the characters to which they may be bound. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ˆchar (e.g. ˆc or ˆu) and ˆ? may be used to indicate Delete. This is very useful for overriding the default terminal settings without having to do an *stty* every time an **xterm** is started.

utmpInhibit (class **UtmpInhibit)**
Specifies whether or not **xterm** should try to record the user's terminal in */etc/utmp*.

sunFunctionKeys (class **SunFunctionKeys)**
Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

waitForMap (class **WaitForMap)**
Specifies whether or not **xterm** should wait for the initial window map before starting the subprocess. The default is ''false.''

The following resources are specified as part of the *vt100* widget (class *VT100*):

allowSendEvents (class **AllowSendEvents)**
Specifies whether or not synthetic key and button events (generated using the X protocol SendEvent request) should be interpreted or discarded. The default is ''false'' meaning they are discarded. Note that allowing such events creates a very large security hole.

alwaysHighlight (class **AlwaysHighlight)**
Specifies whether or not **xterm** should always display a highlighted text cursor. By default, a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus.

appcursorDefault (class **AppcursorDefault)**

If ‘‘true,’’ the cursor keys are initially in application mode. The default is ‘‘false.’’

appkeypadDefault (class **AppkeypadDefault)**
If ‘‘true,’’ the keypad keys are initially in application mode. The default is ‘‘false.’’

autoWrap (class **AutoWrap)**
Specifies whether or not auto-wraparound should be enabled. The default is ‘‘true.’’

bellSuppressTime (class **BellSuppressTime)**
Number of milliseconds after a bell command is sent during which additional bells will be suppressed. Default is 200. If set non-zero, additional bells will also be suppressed until the server reports that processing of the first bell has been completed; this feature is most useful with the visible bell.

boldFont (class **BoldFont)**
Specifies the name of the bold font to use instead of overstriking.

c132 (class **C132)**
Specifies whether or not the VT102 DECCOLM escape sequence should be honored. The default is ‘‘false.’’

charClass (class **CharClass)**
Specifies comma-separated lists of character class bindings of the form [*low-*]*high*:*value*. These are used in determining which sets of characters should be treated the same when doing cut and paste. See the section on specifying character classes.

curses (class **Curses)**
Specifies whether or not the last column bug in *curses*(3x) should be worked around. The default is ‘‘false.’’

background (class **Background)**
Specifies the color to use for the background of the window. The default is ‘‘white.’’

foreground (class **Foreground)**
Specifies the color to use for displaying text in the window. Setting the class name instead of the instance name is an easy way to have everything that would normally appear in the "text" color change color. The default is ‘‘black.’’

cursorColor (class **Foreground)**
Specifies the color to use for the text cursor. The default is ‘‘black.’’

eightBitInput (class **EightBitInput**)
If true, Meta characters input from the keyboard are presented as a single character with the eighth bit turned on. If false, Meta characters are converted into a two-character sequence with the character itself preceded by ESC. The default is ‘‘true.’’

eightBitOutput (class **EightBitOutput**)
Specifies whether or not eight-bit characters sent from the host should be

accepted as is or stripped when printed. The default is ‘‘true.’’

font (class **Font)**
Specifies the name of the normal font. The default is ‘‘fixed.’’

font1 (class **Font1)**
Specifies the name of the first alternative font.

font2 (class **Font2)**
Specifies the name of the second alternative font.

font3 (class **Font3)**
Specifies the name of the third alternative font.

font4 (class **Font4)**
Specifies the name of the fourth alternative font.

font5 (class **Font5)**
Specifies the name of the fifth alternative font.

font6 (class **Font6)**
Specifies the name of the sixth alternative font.

geometry (class **Geometry)**
Specifies the preferred size and position of the VT102 window.

internalBorder (class **BorderWidth)**
Specifies the number of pixels between the characters and the window border. The default is 2.

jumpScroll (class **JumpScroll)**
Specifies whether or not jump scroll should be used. The default is ‘‘true.’’

logFile (class **Logfile)**
Specifies the name of the file to which a terminal session is logged. The default is ‘‘**XtermLog.***XXXXX***’’ (where** *XXXXX* **is the process id of xterm**).

logging (class **Logging)**
Specifies whether or not a terminal session should be logged. The default is ‘‘false.’’

logInhibit (class **LogInhibit)**
Specifies whether or not terminal session logging should be inhibited. The default is ‘‘false.’’

loginShell (class **LoginShell)**
Specifies whether or not the shell to be run in the window should be started as a login shell. The default is ‘‘false.’’

marginBell (class **MarginBell)**
Specifies whether or not the bell should be run when the user types near the right margin. The default is ‘‘false.’’

multiClickTime (class **MultiClickTime)**
Specifies the maximum time in milliseconds between multi-click select events. The default is 250 milliseconds.

multiScroll (class **MultiScroll)**
Specifies whether or not scrolling should be done asynchronously. The default is ‘‘false.’’

nMarginBell (class **Column)**
Specifies the number of characters from the right margin at which the margin bell should be rung, when enabled.

pointerColor (class **Foreground)**
Specifies the foreground color of the pointer. The default is ‘‘XtDefaultForeground.’’

pointerColorBackground (class **Background)**
Specifies the background color of the pointer. The default is ‘‘XtDefaultBackground.’’

pointerShape (class **Cursor)**
Specifies the name of the shape of the pointer. The default is ‘‘xterm.’’

resizeGravity (class **ResizeGravity)**
Affects the behavior when the window is resized to be taller or shorter. **NorthWest** specifies that the top line of text on the screen stay fixed. If the window is made shorter, lines are dropped from the bottom; if the window is made taller, blank lines are added at the bottom. This is compatible with the behavior in R4. **SouthWest** (the default) specifies that the bottom line of text on the screen stay fixed. If the window is made taller, additional saved lines will be scrolled down onto the screen; if the window is made shorter, lines will be scrolled off the top of the screen, and the top saved lines will be dropped.

reverseVideo (class **ReverseVideo)**
Specifies whether or not reverse video should be simulated. The default is ‘‘false.’’

reverseWrap (class **ReverseWrap)**
Specifies whether or not reverse-wraparound should be enabled. The default is ‘‘false.’’

saveLines (class **SaveLines)**
Specifies the number of lines to save beyond the top of the screen when a scrollbar is turned on. The default is 64.

scrollBar (class **ScrollBar)**
Specifies whether or not the scrollbar should be displayed. The default is ‘‘false.’’

scrollTtyOutput (class **ScrollCond)**
Specifies whether or not output to the terminal should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is ‘‘true.’’

scrollKey (class **ScrollCond)**
Specifies whether or not pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is ‘‘false.’’

scrollLines (class **ScrollLines)**

Specifies the number of lines that the *scroll-back* and *scroll-forw* actions should use as a default. The default value is 1.

signalInhibit (class **SignalInhibit)**
Specifies whether or not the entries in the ''Main Options'' menu for sending signals to **xterm** should be disallowed. The default is ''false.''

tekGeometry (class **Geometry)**
Specifies the preferred size and position of the Tektronix window.

tekInhibit (class **TekInhibit)**
Specifies whether or not Tektronix mode should be disallowed. The default is ''false.''

tekSmall (class **TekSmall)**
Specifies whether or not the Tektronix mode window should start in its smallest size if no explicit geometry is given. This is useful when running **xterm** on displays with small screens. The default is ''false.''

tekStartup (class **TekStartup)**
Specifies whether or not **xterm** should start up in Tektronix mode. The default is ''false.''

titeInhibit (class **TiteInhibit)**
Specifies whether or not **xterm** should remove remove *ti* and *te* terminfo entries (used to switch between alternate screens on startup of many screen-oriented programs) from the TERM string. If set, **xterm** also ignores the escape sequence to switch to the alternate screen.

translations (class **Translations)**
Specifies the key and button bindings for menus, selections, ''programmed strings,'' etc. See **ACTIONS** below.

visualBell (class **VisualBell)**
Specifies whether or not a visible bell (i.e. flashing) should be used instead of an audible bell when Control-G is received. The default is ''false.''

The following resources are specified as part of the *tek4014* widget (class *Tek4014*):

width (class **Width)**
Specifies the width of the Tektronix window in pixels.

height (class **Height)**
Specifies the height of the Tektronix window in pixels.

fontLarge (class **Font)**
Specifies the large font to use in the Tektronix window.

font2 (class **Font)**
Specifies font number 2 to use in the Tektronix window.

font3 (class **Font)**
Specifies font number 3 to use in the Tektronix window.

fontSmall (class **Font)**

Specifies the small font to use in the Tektronix window.

initialFont (class **InitialFont)**
Specifies which of the four Tektronix fonts to use initially. Values are the same as for the *set-tek-text* action. The default is ‘‘large.’’

ginTerminator (class **GinTerminator)**
Specifies what character(s) should follow a GIN report or status report. The possibilities are ‘‘none,’’ which sends no terminating characters, ‘‘CRonly,’’ which sends CR, and ‘‘CR&EOT,’’ which sends both CR and EOT. The default is ‘‘none.’’

The resources that may be specified for the various menus are described in the documentation for the Athena **SimpleMenu** widget. The name and classes of the entries in each of the menus are listed below.

The *mainMenu* has the following entries:

securekbd (class **SmeBSB)**
This entry invokes the **secure()** action.

allowsends (class **SmeBSB)**
This entry invokes the **allow-send-events(toggle)** action.

logging (class **SmeBSB)**
This entry invokes the **set-logging(toggle)** action.

redraw (class **SmeBSB)**
This entry invokes the **redraw()** action.

line1 (class **SmeLine)**
This is a separator.

suspend (class **SmeBSB)**
This entry invokes the **send-signal(tstp)** action on systems that support job control.

continue (class **SmeBSB)**
This entry invokes the **send-signal(cont)** action on systems that support job control.

interrupt (class **SmeBSB)**
This entry invokes the **send-signal(int)** action.

hangup (class **SmeBSB)**
This entry invokes the **send-signal(hup)** action.

terminate (class **SmeBSB)**
This entry invokes the **send-signal(term)** action.

kill (class **SmeBSB)**
This entry invokes the **send-signal(kill)** action.

line2 (class **SmeLine)**
This is a separator.

quit (class **SmeBSB)**
: This entry invokes the **quit()** action.

The *vtMenu* has the following entries:

scrollbar (class **SmeBSB)**
: This entry invokes the **set-scrollbar(toggle)** action.

jumpscroll (class **SmeBSB)**
: This entry invokes the **set-jumpscroll(toggle)** action.

reversevideo (class **SmeBSB)**
: This entry invokes the **set-reverse-video(toggle)** action.

autowrap (class **SmeBSB)**
: This entry invokes the **set-autowrap(toggle)** action.

reversewrap (class **SmeBSB)**
: This entry invokes the **set-reversewrap(toggle)** action.

autolinefeed (class **SmeBSB)**
: This entry invokes the **set-autolinefeed(toggle)** action.

appcursor (class **SmeBSB)**
: This entry invokes the **set-appcursor(toggle)** action.

appkeypad (class **SmeBSB)**
: This entry invokes the **set-appkeypad(toggle)** action.

scrollkey (class **SmeBSB)**
: This entry invokes the **set-scroll-on-key(toggle)** action.

scrollttyoutput (class **SmeBSB)**
: This entry invokes the **set-scroll-on-tty-output(toggle)** action.

allow132 (class **SmeBSB)**
: This entry invokes the **set-allow132(toggle)** action.

cursesemul (class **SmeBSB)**
: This entry invokes the **set-cursesemul(toggle)** action.

visualbell (class **SmeBSB)**
: This entry invokes the **set-visualbell(toggle)** action.

marginbell (class **SmeBSB)**
: This entry invokes the **set-marginbell(toggle)** action.

altscreen (class **SmeBSB)**
: This entry is currently disabled.

line1 (class **SmeLine)**
: This is a separator.

softreset (class **SmeBSB)**
: This entry invokes the **soft-reset()** action.

hardreset (class **SmeBSB)**
: This entry invokes the **hard-reset()** action.

clearsavedlines (class **SmeBSB)**
This entry invokes the **clear-saved-lines()** action.

line2 (class **SmeLine)**
This is a separator.

tekshow (class **SmeBSB)**
This entry invokes the **set-visibility(tek,toggle)** action.

tekmode (class **SmeBSB)**
This entry invokes the **set-terminal-type(tek)** action.

vthide (class **SmeBSB)**
This entry invokes the **set-visibility(vt,off)** action.

The *fontMenu* has the following entries:

fontdefault (class **SmeBSB)**
This entry invokes the **set-vt-font(d)** action.

font1 (class **SmeBSB)**
This entry invokes the **set-vt-font(1)** action.

font2 (class **SmeBSB)**
This entry invokes the **set-vt-font(2)** action.

font3 (class **SmeBSB)**
This entry invokes the **set-vt-font(3)** action.

font4 (class **SmeBSB)**
This entry invokes the **set-vt-font(4)** action.

font5 (class **SmeBSB)**
This entry invokes the **set-vt-font(5)** action.

font6 (class **SmeBSB)**
This entry invokes the **set-vt-font(6)** action.

fontescape (class **SmeBSB)**
This entry invokes the **set-vt-font(e)** action.

fontsel (class **SmeBSB)**
This entry invokes the **set-vt-font(s)** action.

The *tekMenu* has the following entries:

tektextlarge (class **SmeBSB)**
This entry invokes the **set-tek-text(l)** action.

tektext2 (class **SmeBSB)**
This entry invokes the **set-tek-text(2)** action.

tektext3 (class **SmeBSB)**
This entry invokes the **set-tek-text(3)** action.

tektextsmall (class **SmeBSB)**
This entry invokes the **set-tek-text(s)** action.

line1 (class **SmeLine)**
This is a separator.

tekpage (class **SmeBSB)**
This entry invokes the **tek-page()** action.

tekreset (class **SmeBSB)**
This entry invokes the **tek-reset()** action.

tekcopy (class **SmeBSB)**
This entry invokes the **tek-copy()** action.

line2 (class **SmeLine)**
This is a separator.

vtshow (class **SmeBSB)**
This entry invokes the **set-visibility(vt,toggle)** action.

vtmode (class **SmeBSB)**
This entry invokes the **set-terminal-type(vt)** action.

tekhide (class **SmeBSB)**
This entry invokes the **set-visibility(tek,toggle)** action.

The following resources are useful when specified for the Athena Scrollbar widget:

thickness (class **Thickness)**
Specifies the width in pixels of the scrollbar.

background (class **Background)**
Specifies the color to use for the background of the scrollbar.

foreground (class **Foreground)**
Specifies the color to use for the foreground of the scrollbar. The ‘‘thumb’’ of the scrollbar is a simple checkerboard pattern alternating pixels for foreground and background color.

POINTER USAGE

Once the VT102 window is created, *xterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the ‘‘shift’’ key. The assignment of the functions described below to keys and buttons may be changed through the resource database; see **ACTIONS** below.

Pointer button one (usually left) is used to save text into the cut buffer. Move the cursor to beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Quadruple-clicking goes back to characters, etc. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key / button bindings specify that an X selection is to be made, **xterm** will leave the selected text highlighted for as long as it is the selection owner.

Pointer button two (usually middle) 'types' (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. (Without loss of generality, you can swap ''right'' and ''left'' everywhere in the rest of this paragraph.) If pressed while closer to the right edge of the selection than the left, it extends/contracts the right edge of the selection. If you contract the selection past the left edge of the selection, *xterm* assumes you really meant the left edge, restores the original selection, then extends/contracts the left edge of the selection. Extension starts in the selection unit mode that the last selection or extension was performed in; you can multiple-click to cycle through them.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a 'file' whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e., the text is delimited by new lines.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved. As more text is saved (up to the maximum), the size of the highlighted area decreases.

Clicking button one with the pointer in the scroll region moves the adjacent line to the top of the display window.

Clicking button three moves the top line of the display window down to the pointer position.

Clicking button two moves the display to a position in the saved text that corresponds to the pointer's position in the scrollbar.

Unlike the VT102 window, the Tektronix window dows not allow the copying of text. It does allow Tektronix GIN mode, and in this mode the cursor will change from an arrow to a cross. Pressing any key will send that key and the current coordinate of the cross cursor. Pressing button one, two, or three will return the letters 'l', 'm', and 'r', respectively. If the 'shift' key is pressed when a pointer button is pressed, the corresponding upper case letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this is bit is normally stripped unless the terminal mode is RAW; see **tty**(1) and **tty**(7) for details).

MENUS

Xterm has four menus, named *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*. Each menu pops up under the correct combinations of key and button presses. Most menus are divided into two section, separated by a horizontal line. The top portion contains various modes that can be altered. A check mark appears next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion of the menu are command entries; selecting one of these performs the indicated function.

The **xterm** menu pops up when the ''control'' key and pointer button one are pressed in a window. The *mainMenu* contains items that apply to both the VT102 and Tektronix windows. The **Secure Keyboard** mode is be used when typing in passwords or other sensitive data in an unsecure environment; see **SECURITY** below. Notable entries in the command section of the menu are the **Continue**, **Suspend**, **Interrupt**, **Hangup**, **Terminate** and **Kill** which sends the SIGCONT, SIGTSTP, SIGINT, SIGHUP, SIGTERM and SIGKILL signals, respectively, to the process group of the process running under *xterm* (usually the shell). The **Continue** function is especially useful if the user has accidentally typed CTRL-Z, suspending the process.

The *vtMenu* sets various modes in the VT102 emulation, and is popped up when the ''control'' key and pointer button two are pressed in the VT102 window. In the command section of this menu, the soft reset entry will reset scroll regions. This can be convenient when some program has left the scroll regions set incorrectly (often a problem when using VMS or TOPS-20). The full reset entry will clear the screen, reset tabs to every eight columns, and reset the terminal modes (such as wrap and smooth scroll) to their initial states just after *xterm* has finished processing the command line options.

The *fontMenu* sets the font used in the VT102 window. In addition to the default font and a number of alternatives that are set with resources, the menu offers the font last specified by the Set Font escape sequence (see the document *Xterm Control Sequences*) and the current selection as a font name (if the PRIMARY selection is owned).

The *tekMenu* sets various modes in the Tektronix emulation, and is popped up when the ''control'' key and pointer button two are pressed in the Tektronix window. The current font size is checked in the modes section of the menu. The **PAGE** entry in the command section clears the Tektronix window.

SECURITY

X environments differ in their security consciousness. MIT servers, run under *xdm*, are capable of using a ''magic cookie'' authorization scheme that can provide a reasonable level of security for many people. If your server is only using a host-based mechanism to control access to the server (see **xhost**(1)), then if you enable access for a host and other users are also permitted to run clients on that same host, there is every possibility that someone can run an application that will use the basic services of the X protocol to snoop on your activities, potentially capturing a transcript of everything you type at the keyboard. This is of particular concern when you want to type in a password or other sensitive data. The best solution to this problem is to use a better authorization mechanism that host-based control, but a simple mechanism exists for protecting keyboard input in **xterm**.

The **xterm** menu (see **MENUS** above) contains a **Secure Keyboard** entry which, when enabled, ensures that all keyboard input is directed *only* to **xterm** (using the GrabKeyboard protocol request). When an application prompts you for a password (or other sensitive data), you can enable **Secure Keyboard** using the menu, type in the data, and then disable **Secure Keyboard** using the menu again. Only one X client at a time can secure the keyboard, so when you attempt to enable **Secure Keyboard** it may fail. In this case, the bell will sound. If the **Secure Keyboard** succeeds, the foreground and background colors will be exchanged (as if you selected the **Reverse Video** entry in the **Modes** menu); they will be exchanged again when you exit secure mode. If the colors do *not* switch, then you should be *very* suspicious that you are being spoofed. If the application you are running displays a prompt before asking for the password, it is safest to enter secure mode *before* the prompt gets displayed, and to make sure that the prompt gets displayed correctly (in the new colors), to minimize the probability of spoofing. You can also bring up the menu again and make sure that a check mark appears next to the entry.

Secure Keyboard mode will be disabled automatically if your xterm window becomes iconified (or otherwise unmapped), or if you start up a reparenting window manager (that places a title bar or other decoration around the window) while in **Secure Keyboard** mode. (This is a feature of the X protocol not easily overcome.) When this happens, the foreground and background colors will be switched back and the bell will sound in warning.

CHARACTER CLASSES

Clicking the middle mouse button twice in rapid succession will cause all characters of the same class (e.g. letters, white space, punctuation) to be selected. Since different people have different preferences for what should be selected (for example, should filenames be selected as a whole or only the separate subnames), the default mapping can be overridden through the use of the *charClass* (class *CharClass*) resource.

This resource is simply a list of *range*:*value* pairs where the range is either a single number or *low-high* in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The *value* is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is:

```
static int charClass[128] = {
/* NUL  SOH  STX  ETX  EOT  ENQ  ACK  BEL */
   32,   1,   1,   1,   1,   1,   1,   1,
/*  BS   HT   NL   VT   NP   CR   SO   SI */
    1,  32,   1,   1,   1,   1,   1,   1,
/* DLE  DC1  DC2  DC3  DC4  NAK  SYN  ETB */
    1,   1,   1,   1,   1,   1,   1,   1,
/* CAN   EM  SUB  ESC   FS   GS   RS   US */
    1,   1,   1,   1,   1,   1,   1,   1,
/*  SP    !    "    #    $    %    &    ' */
   32,  33,  34,  35,  36,  37,  38,  39,
/*   (    )    *    +    ,    -    .    / */
   40,  41,  42,  43,  44,  45,  46,  47,
/*   0    1    2    3    4    5    6    7 */
   48,  48,  48,  48,  48,  48,  48,  48,
/*   8    9    :    ;    <    =    >    ? */
   48,  48,  58,  59,  60,  61,  62,  63,
/*   @    A    B    C    D    E    F    G */
   64,  48,  48,  48,  48,  48,  48,  48,
/*   H    I    J    K    L    M    N    O */
   48,  48,  48,  48,  48,  48,  48,  48,
/*   P    Q    R    S    T    U    V    W */
   48,  48,  48,  48,  48,  48,  48,  48,
/* X   Y   Z   [   \   ]   ^   _ */
  48, 48, 48, 91, 92, 93, 94, 48,
/* `   a   b   c   d   e   f   g */
  96, 48, 48, 48, 48, 48, 48, 48,
/* h   i   j   k   l   m   n   o */
  48, 48, 48, 48, 48, 48, 48, 48,
/* p   q   r   s   t   u   v   w */
  48, 48, 48, 48, 48, 48, 48, 48,
/* x   y   z   {   |   }   ~  DEL */
  48, 48, 48,123,124,125,126,  1};
```

For example, the string ‘‘33:48,37:48,45-47:48,64:48’’ indicates that the exclamation mark, percent sign, dash, period, slash, and ampersand characters should be treated the same way as characters and numbers. This is very useful for cutting and pasting electronic mailing addresses and filenames.

ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the vt100 or tek4014 widgets. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for using within the **vt100** or **tek4014 translations** resources:

bell([*percent*])
This action rings the keyboard bell at the specified percentage above or below the base volume.

ignore() This action ignores the event but checks for special pointer position escape sequences.

insert() This action inserts the character or string associated with the key that was pressed.

insert-seven-bit()
This action is a synonym for **insert()**

insert-eight-bit()
This action inserts an eight-bit (Meta) version of the character or string associated with the key that was pressed. The exact action depends on the value of the **eightBitInput** resource.

insert-selection(*sourcename* [, ...])
This action inserts the string found in the selection or cutbuffer indicated by *sourcename*. Sources are checked in the order given (case is significant) until one is found. Commonly-used selections include: *PRIMARY*, *SECONDARY*, and *CLIPBOARD*. Cut buffers are typically named *CUT_BUFFER0* through *CUT_BUFFER7*.

keymap(*name*)
This action dynamically defines a new translation table whose resource name is *name* with the suffix *Keymap* (case is significant). The name *None* restores the

original translation table.

popup-menu(*menuname***)**
This action displays the specified popup menu. Valid names (case is significant) include: *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*.

secure() This action toggles the **Secure Keyboard** mode described in the section named **SECURITY**, and is invoked from the **securekbd** entry in *mainMenu*.

select-start()
This action begins text selection at the current pointer location. See the section on **POINTER USAGE** for information on making selections.

select-extend()
This action tracks the pointer and extends the selection. It should only be bound to Motion events.

select-end(*destname* **[, ...])**
This action puts the currently selected text into all of the selections or cutbuffers specified by *destname*.

select-cursor-start()
This action is similar to **select-start** except that it begins the selection at the current text cursor position.

select-cursor-end(*destname* **[, ...])**
This action is similar to **select-end** except that it should be used with **select-cursor-start**.

set-vt-font(*d/1/2/3/4/5/6/e/s* **[,***normalfont* **[,** *boldfont***]])**
This action sets the font or fonts currently being used in the VT102 window. The first argument is a single character that specifies the font to be used: *d* or *D* indicate the default font (the font initially used when **xterm** was started), *1* through *6* indicate the fonts specified by the *font1* through *font6* resources, *e* or *E* indicate the normal and bold fonts that have been set through escape codes (or specified as the second and third action arguments, respectively), and *s* or *S* indicate the font selection (as made by programs such as *xfontsel(1)*) indicated by the second action argument.

start-extend()
This action is similar to **select-start** except that the selection is extended to the current pointer location.

start-cursor-extend()
This action is similar to **select-extend** except that the selection is extended to the current text cursor position.

string(*string***)**
This action inserts the specified text string as if it had been typed. Quotation is necessary if the string contains whitespace or non-alphanumeric characters. If the string argument begins with the characters ‘‘0x’’, it is interpreted as a hex character constant.

scroll-back(*count* **[,***units***])**

This action scrolls the text window backward so that text that had previously scrolled off the top of the screen is now visible. The *count* argument indicates the number of *units* (which may be *page*, *halfpage*, *pixel*, or *line*) by which to scroll.

scroll-forw(*count* **[,***units***])**
: This action scrolls is similar to **scroll-back** except that it scrolls the other direction.

allow-send-events(*on/off/toggle***)**
: This action set or toggles the **allowSendEvents** resource and is also invoked by the *allowsends* entry in *mainMenu*.

set-logging(*on/off/toggle***)**
: This action toggles the **logging** resource and is also invoked by the **logging** entry in *mainMenu*.

redraw()
: This action redraws the window and is also invoked by the *redraw* entry in *mainMenu*.

send-signal(*signame***)**
: This action sends the signal named by **signame** to the **xterm** subprocess (the shell or program specified with the *–e* command line option) and is also invoked by the **suspend**, **continue**, **interrupt**, **hangup**, **terminate**, and *kill* entries in *mainMenu*. Allowable signal names are (case is not significant): *tstp* (if supported by the operating system), *suspend* (same as *tstp*), *cont* (if supported by the operating system), *int*, *hup*, *term*, *quit*, *alrm*, *alarm* (same as *alrm*) and *kill*.

quit() This action sends a SIGHUP to the subprogram and exits. It is also invoked by the **quit** entry in *mainMenu*.

set-scrollbar(*on/off/toggle***)**
: This action toggles the **scrollbar** resource and is also invoked by the **scrollbar** entry in *vtMenu*.

set-jumpscroll(*on/off/toggle***)**
: This action toggles the **jumpscroll** resource and is also invoked by the **jumpscroll** entry in *vtMenu*.

set-reverse-video(*on/off/toggle***)**
: This action toggles the **reverseVideo** resource and is also invoked by the **rever-***sevideo* entry in *vtMenu*.

set-autowrap(*on/off/toggle***)**
: This action toggles automatic wrapping of long lines and is also invoked by the **autowrap** entry in *vtMenu*.

set-reversewrap(*on/off/toggle***)**
: This action toggles the **reverseWrap** resource and is also invoked by the **rever-***sewrap* entry in *vtMenu*.

set-autolinefeed(*on/off/toggle***)**
: This action toggles automatic insertion of linefeeds and is also invoked by the

autolinefeed entry in *vtMenu*.

set-appcursor(*on/off/toggle***)**
: This action toggles the handling Application Cursor Key mode and is also invoked by the Bappcursor entry in *vtMenu*.

set-appkeypad(*on/off/toggle***)**
: This action toggles the handling of Application Keypad mode and is also invoked by the **appkeypad** entry in *vtMenu*.

set-scroll-on-key(*on/off/toggle***)**
: This action toggles the **scrollKey** resource and is also invoked from the **scrollkey** entry in *vtMenu*.

set-scroll-on-tty-output(*on/off/toggle***)**
: This action toggles the **scrollTtyOutput** resource and is also invoked from the *scrollttyoutput* entry in *vtMenu*.

set-allow132(*on/off/toggle***)**
: This action toggles the **c132** resource and is also invoked from the **allow132** entry in *vtMenu*.

set-cursesemul(*on/off/toggle***)**
: This action toggles the **curses** resource and is also invoked from the **cursesemul** entry in *vtMenu*.

set-visual-bell(*on/off/toggle***)**
: This action toggles the **visualBell** resource and is also invoked by the **visualbell** entry in *vtMenu*.

set-marginbell(*on/off/toggle***)**
: This action toggles the **marginBell** resource and is also invoked from the **margin-***bell* entry in *vtMenu*.

set-altscreen(*on/off/toggle***)**
: This action toggles between the alternate and current screens.

soft-reset()
: This action resets the scrolling region and is also invoked from the **softreset** entry in *vtMenu*.

hard-reset()
: This action resets the scrolling region, tabs, window size, and cursor keys and clears the screen. It is also invoked from the **hardreset** entry in *vtMenu*.

clear-saved-lines()
: This action does **hard-reset()** (see above) and also clears the history of lines saved off the top of the screen. It is also invoked from the **clearsavedlines** entry in *vtMenu*.

set-terminal-type(*type***)**
: This action directs output to either the *vt* or *tek* windows, according to the *type* string. It is also invoked by the **tekmode** entry in *vtMenu* and the **vtmode** entry in *tekMenu*.

set-visibility(*vt/tek***,***on/off/toggle***)**
This action controls whether or not the *vt* or *tek* windows are visible. It is also invoked from the **tekshow** and **vthide** entries in *vtMenu* and the **vtshow** and *tekhide* entries in *tekMenu*.

set-tek-text(*large/2/3/small***)**
This action sets font used in the Tektronix window to the value of the resources **tektextlarge**, **tektext2**, **tektext3**, and **tektextsmall** according to the argument. It is also by the entries of the same names as the resources in *tekMenu*.

tek-page()
This action clears the Tektronix window and is also invoked by the **tekpage** entry in *tekMenu*.

tek-reset()
This action resets the Tektronix window and is also invoked by the *tekreset* entry in *tekMenu*.

tek-copy()
This action copies the escape codes used to generate the current window contents to a file in the current directory beginning with the name COPY. It is also invoked from the *tekcopy* entry in *tekMenu*.

visual-bell()
This action flashes the window quickly.

The Tektronix window also has the following action:

gin-press(*l/L/m/M/r/R***)**
This action sends the indicated graphics input code.

The default bindings in the VT102 window are:

```
Shift <KeyPress> Prior:        scroll-back(1,halfpage) \ n\
Shift <KeyPress> Next:         scroll-forw(1,halfpage) \ n\
Shift <KeyPress> Select:       select-cursor-start() \
                               select-cursor-end(PRIMARY, CUT_BUFFER0) \ n\
Shift <KeyPress> Insert:       insert-selection(PRIMARY, CUT_BUFFER0) \ n\
˜Meta<KeyPress>:               insert-seven-bit() \ n\
Meta<KeyPress>:                insert-eight-bit() \ n\
!Ctrl <Btn1Down>:              popup-menu(mainMenu) \ n\
!Lock Ctrl <Btn1Down>:         popup-menu(mainMenu) \ n\
˜Meta <Btn1Down>:              select-start() \ n\
˜Meta <Btn1Motion>:            select-extend() \ n\
!Ctrl <Btn2Down>:              popup-menu(vtMenu) \ n\
!Lock Ctrl <Btn2Down>:         popup-menu(vtMenu) \ n\
˜Ctrl ˜Meta <Btn2Down>:        ignore() \ n\
˜Ctrl ˜Meta <Btn2Up>:          insert-selection(PRIMARY, CUT_BUFFER0) \ n\
!Ctrl <Btn3Down>:              popup-menu(fontMenu) \ n\
!Lock Ctrl <Btn3Down>:         popup-menu(fontMenu) \ n\
˜Ctrl ˜Meta <Btn3Down>:        start-extend() \ n\
˜Meta <Btn3Motion>:          select-extend() \ n\
<BtnUp>:                     select-end(PRIMARY, CUT_BUFFER0) \ n\
<BtnDown>:                   bell(0)
```

The default bindings in the Tektronix window are:

```
        ˜Meta<KeyPress>:             insert-seven-bit() \ n\
        Meta<KeyPress>:              insert-eight-bit() \ n\
        !Ctrl <Btn1Down>:            popup-menu(mainMenu) \ n\
        !Lock Ctrl <Btn1Down>:       popup-menu(mainMenu) \ n\
        !Ctrl <Btn2Down>:            popup-menu(tekMenu) \ n\
        !Lock Ctrl <Btn2Down>:       popup-menu(tekMenu) \ n\
        Shift ˜Meta<Btn1Down>:       gin-press(L) \ n\
        ˜Meta<Btn1Down>:             gin-press(l) \ n\
        Shift ˜Meta<Btn2Down>:       gin-press(M) \ n\
        ˜Meta<Btn2Down>:             gin-press(m) \ n\
        Shift ˜Meta<Btn3Down>:       gin-press(R) \ n\
        ˜Meta<Btn3Down>:             gin-press(r)
```

Below is a sample how of the **keymap()** action is used to add special keys for entering commonly-typed works:

```
        *VT100.Translations: #override <Key>F13: keymap(dbx)
        *VT100.dbxKeymap.translations: \
                <Key>F14:     keymap(None) \ n\
                <Key>F17:     string("next") string(0x0d) \ n\
                <Key>F18:     string("step") string(0x0d) \ n\
                <Key>F19:     string("continue") string(0x0d) \ n\
                <Key>F20:     string("print ") insert-selection(PRIMARY, CUT_BUFFER0)
```

ENVIRONMENT

Xterm sets the environment variable ‘‘TERM’’ properly for the size window you have created. It also uses and sets the environment variable ‘‘DISPLAY’’ to specify which bit map display terminal to use. The environment variable ‘‘WINDOWID’’ is set to the X window id number of the **xterm** window.

SEE ALSO

resize(1), **X11**(7), **tty**(1)

BUGS

Large pastes do not work on some systems. This is not a bug in **xterm** ; it is a bug in the pseudo terminal driver of those systems. **xterm feeds large pastes to the** will accept data, but some pty drivers do not return enough information to know if the write has succeeded.

Many of the options are not resettable after **xterm** starts.

The Tek widget does not support key / button re-binding.

Only fixed-width, character-cell fonts are supported.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that don't know about each other. Ideally, you'd like to be able to pick and choose emulator widgets and stick them into a single control widget.

There needs to be a dialog box to allow entry of log file name and the COPY file name.

COPYRIGHT

AUTHORS

Far too many people, including:

Loretta Guarino Reid (DEC-UEG-WSL), Joel McCormack (DEC-UEG-WSL), Terry Weissman (DEC-UEG-WSL), Edward Moy (Berkeley), Ralph R. Swick (MIT-Athena), Mark Vandevoorde (MIT-Athena), Bob McNamara (DEC-MAD), Jim Gettys (MIT-Athena), Bob Scheifler (MIT X Consortium), Doug Mink (SAO), Steve Pitschke (Stellar), Ron Newman (MIT-Athena), Jim Fulton (MIT X Consortium), Dave Serisky (HP), Jonathan Kamens (MIT-Athena)

NAME

xwd – dump an image of an X window

SYNOPSIS

xwd [-debug] [-help] [-nobdrs] [-out *file*] [-xy] [-frame] [-add *value*] [-root | -id *id* | -name *name*] [-icmap] [-screen] [-display *display*]

DESCRIPTION

Xwd is an X Window System window dumping utility. **Xwd** allows X users to store window images in a specially formatted dump file. This file can then be read by various other X utilities for redisplay, printing, editing, formatting, archiving, image processing, etc. The target window is selected by clicking the pointer in the desired window. The keyboard bell is rung once at the beginning of the dump and twice when the dump is completed.

OPTIONS

-display *display*
This argument allows you to specify the server to connect to; see **X**(7)

-help Print out the ‘Usage:’ command syntax summary.

-nobdrs This argument specifies that the window dump should not include the pixels that compose the X window border. This is useful in situations where you may wish to include the window contents in a document as an illustration.

-out *file* This argument allows the user to explicitly specify the output file on the command line. The default is to output to standard out.

-xy This option applies to color displays only. It selects ‘XY’ format dumping instead of the default ‘Z’ format.

-add *value*
This option specifies an signed value to be added to every pixel.

-frame This option indicates that the window manager frame should be included when manually selecting a window.

-root This option indicates that the root window should be selected for the window dump, without requiring the user to select a window with the pointer.

-id *id* This option indicates that the window with the specified resource id should be selected for the window dump, without requiring the user to select a window with the pointer.

-name *name*
This option indicates that the window with the specified WM_NAME property should be selected for the window dump, without requiring the user to select a window with the pointer.

-icmap Normally the colormap of the chosen window is used to obtain RGB values. This option forces the first installed colormap of the screen to be used instead.

-screen This option indicates that the GetImage request used to obtain the image should be done on the root window, rather than directly on the specified window. In this way, you can obtain pieces of other windows that overlap the specified window, and more importantly, you can capture menus or other popups that are independent windows but appear over the specified window.

ENVIRONMENT

DISPLAY
To get default host and display number.

CAVEAT

If the dumped window is partially obscured by another window then the obscured area has undefined content.

FILES

XWDFile.h
X Window Dump File format definition file.

SEE ALSO

xwud(1), **xpr**(1), **X11**(7)

COPYRIGHT

AUTHORS

Tony Della Fera, Digital Equipment Corp., MIT Project Athena
William F. Wyatt, Smithsonian Astrophysical Observatory

NAME

xwininfo – window information utility for X

SYNOPSIS

xwininfo [–help] [–id *id*] [–root] [–name *name*] [–int] [–children] [–tree] [–stats] [–bits] [–events] [–size] [–wm] [–shape] [–frame] [–all] [–english] [–metric] [–display *display*]

DESCRIPTION

Xwininfo is a utility for displaying information about windows. Various information is displayed depending on which options are selected. If no options are chosen, **–stats** is assumed.

The user has the option of selecting the target window with the mouse (by clicking any mouse button in the desired window) or by specifying its window id on the command line with the **–id** option. Or instead of specifying the window by its id number, the **–name** option may be used to specify which window is desired by name. There is also a special **–root** option to quickly obtain information on the screen's root window.

OPTIONS

–help Print out the 'Usage:' command syntax summary.

–id *id* This option allows the user to specify a target window *id* on the command line rather than using the mouse to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the mouse might be impossible or interfere with the application.

–name *name*
This option allows the user to specify that the window named *name* is the target window on the command line rather than using the mouse to select the target window.

–root This option specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

–int This option specifies that all X window ids should be displayed as integer values. The default is to display them as hexadecimal values.

–children
This option causes the root, parent, and children windows' ids and names of the selected window to be displayed.

–tree This option is like **–children** but displays all children recursively.

–stats This option causes the display of various attributes pertaining to the location and appearance of the selected window. Information displayed includes the location of the window, its width and height, its depth, border width, class, colormap id if any, map state, backing-store hint, and location of the corners.

–bits This option causes the display of various attributes pertaining to the selected window's raw bits and how the selected window is to be stored. Displayed information includes the selected window's bit gravity, window gravity, backing-store hint, backing-planes value, backing pixel, and whether or not the window has save-under set.

–events This option causes the selected window’s event masks to be displayed. Both the event mask of events wanted by some client and the event mask of events not to propagate are displayed.

–size This option causes the selected window’s sizing hints to be displayed. Displayed information includes: for both the normal size hints and the zoom size hints, the user supplied location if any; the program supplied location if any; the user supplied size if any; the program supplied size if any; the minimum size if any; the maximum size if any; the resize increments if any; and the minimum and maximum aspect ratios if any.

–wm This option causes the selected window’s window manager hints to be displayed. Information displayed may include whether or not the application accepts input, what the window’s icon window # and name is, where the window’s icon should go, and what the window’s initial state should be.

–shape This option causes the selected window’s window and border shape extents to be displayed.

–frame This option causes window manager frames to be considered when manually selecting windows.

–metric This option causes all individual height, width, and x and y positions to be displayed in millimeters as well as number of pixels, based on what the server thinks the resolution is. Geometry specifications that are in **+x+y** form are not changed.

–english
This option causes all individual height, width, and x and y positions to be displayed in inches (and feet, yards, and miles if necessary) as well as number of pixels. **–metric** and **–english** may both be enabled at the same time.

–all This option is a quick way to ask for all information possible.

–display *display*
This option allows you to specify the server to connect to; see *X(1)*.

EXAMPLE

The following is a sample summary taken with no options specified:

```
xwininfo: Window id: 0x60000f "xterm"

  Absolute upper-left X: 2
  Absolute upper-left Y: 85
  Relative upper-left X:  0
  Relative upper-left Y:  25
  Width: 579
  Height: 316
  Depth: 8
  Visual Class: PseudoColor
  Border width: 0
  Class: InputOutput
  Colormap: 0x27 (installed)
  Bit Gravity State: NorthWestGravity
  Window Gravity State: NorthWestGravity
  Backing Store State: NotUseful
  Save Under State: no
  Map State: IsViewable
  Override Redirect State: no
  Corners:  +2+85  -699+85  -699-623  +2-623
  -geometry 80x24+0+58
```

ENVIRONMENT

DISPLAY
To get the default host and display number.

SEE ALSO

X11(7), **xprop**(1)

BUGS

Using –**stats** –**bits** shows some redundant information.

The -geometry string displayed must make assumptions about the window's border width and the behavior of the application and the window manager. As a result, the location given is not always correct.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X11(7)* for a full statement of rights and permissions.

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xwud – image displayer for X

SYNOPSIS

xwud [**-in** *file*] [**-noclick**] [**-geometry** *geom*] [**-display** *display*] [**-new**] [**-std** *maptype*] [**-raw**] [**-vis** *vis-type-or-id*] [**-help**] [**-rv**] [**-plane** *number*] [**-fg** *color*] [**-bg** *color*]

DESCRIPTION

Xwud is an X Window System image undumping utility. **Xwud** allows X users to display in a window an image saved in a specially formatted dump file, such as produced by **xwd**(1).

OPTIONS

-bg *color*
If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the "0" bits in the image.

-display *display*
This option allows you to specify the server to connect to; see **X11**(1).

-fg *color*
If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the "1" bits in the image.

-geometry *geom*
This option allows you to specify the size and position of the window. Typically you will only want to specify the position, and let the size default to the actual size of the image.

-help
Print out a short description of the allowable options.

-in *file*
This option allows the user to explicitly specify the input file on the command line. If no input file is given, the standard input is assumed.

-new
This option forces creation of a new colormap for displaying the image. If the image characteristics happen to match those of the display, this can get the image on the screen faster, but at the cost of using a new colormap (which on most displays will cause other windows to go technicolor).

-noclick
Clicking any button in the window will terminate the application, unless this option is specified. Termination can always be achieved by typing ’q’, ’Q’, or ctrl-c.

-plane *number*
You can select a single bit plane of the image to display with this option. Planes are numbered with zero being the least significant bit. This option can be used to figure out which plane to pass to **xpr**(1) for printing.

-raw
This option forces the image to be displayed with whatever color values happen to currently exist on the screen. This option is mostly useful when undumping an image back onto the same screen that the image originally came from, while the original windows are still on the screen, and results in getting the image on the screen faster.

-rv
If a bitmap image (or a single plane of an image) is displayed, this option forces the foreground and background colors to be swapped. This may be needed when displaying a bitmap image which has the color sense of pixel values "0"

and "1" reversed from what they are on your display.

-std *maptype*
This option causes the image to be displayed using the specified Standard Colormap. The property name is obtained by converting the type to upper case, prepending "RGB_", and appending "_MAP". Typical types are "best", "default", and "gray". See **xstdcmap**(1) for one way of creating Standard Colormaps.

-vis *vis-type-or-id*
This option allows you to specify a particular visual or visual class. The default is to pick the "best" one. A particular class can be specified: "StaticGray", "GrayScale", "StaticColor", "PseudoColor", "DirectColor", or "TrueColor". Or "Match" can be specified, meaning use the same class as the source image. Alternatively, an exact visual id (specific to the server) can be specified, either as a hexadecimal number (prefixed with "0x") or as a decimal number. Finally, "default" can be specified, meaning to use the same class as the colormap of the root window. Case is not significant in any of these strings.

ENVIRONMENT

DISPLAY
To get default display.

FILES

XWDFile.h
X Window Dump File format definition file.

SEE ALSO

xwd(1), **xpr**(1), **xstdcmap**(1), **X11**(7)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See **X11**(7) for a full statement of rights and permissions.

AUTHOR

Bob Scheifler, MIT X Consortium

Index

A

accessx(1) — interface to keyboard enhancements, 1
appres(1) — application resource database utility, 1
atobm(1) — ASCII to bitmap conversion utility, 1

B

bdftopcf(1) — font conversion utility, 1
bdftosnf(1) — font conversion utility, 1
bitmap(1) — bitmap editor & conversion utilities, 1
bldrgb(1) — colorname database utility, 1
bmtoa(1) — bitmap to ASCII conversion utility, 1

C

cmap_alloc(1) — colormap utility, 1
cmap_compact(1) — utility to reduce colormap flashing, 1
color
- building the colorname database — **rgb**(1), 1
- characterization data — **xcmsdb**(1), 1
- colorname database utility — **bldrgb**(1), 1
- default colormap allocation — **cmap_alloc**(1), 1
- displaying the colorname database — **showrgb**(1), 1
- display current colors — **xcolor**(1), 1
- preventing flashing — **cmap_compact**(1), 1
- standard colormap properties — **xstdcmap**(1), 1
- swap red & blue — **redxblue**(1), 1

constype(1) — prints console type, 1
cps(1) — PostScript language interface, 1

D

demonstration programs
- command-line interface for PostScript — **dpsexec**(6), 1
- draws plaid patterns — **plaid**(6), 1
- graphics primitives demo — **xgc**(6), 1
- improved polyhedron demo — **ico2**(6), 1
- MacPaint display demo — **xmac**(6), 1
- maze demo — **maze**(6), 1
- pattern drawing demo — **muncher**(6), 1
- puzzle demo — **puzzle**(1), 1
- rotating polyhedron — **ico**(6), 1
- screen magnification demo — **xmag_multivis**(6), 1
- solitaire game demo — **xsol**(6), 1
- spinning text game — **texteroids**(6), 1
- worm drawing demo — **worms**(6), 1

dps(7) — general information about Display Postscript, 1
dpsexec(6) — command-line interface for PostScript, 1

E

editres(1) — X resource editor, 1

F

font server
- administration utility — **fsadmin**(1), 1
- BDF font generator — **fstobdf**(1), 1
- command options — **fs**(1), 1
- font dumper — **showfont**(1), 1
- font list displayer — **fslsfonts**(1), 1
- information utility — **fsinfo**(1), 1

fonts
- bitmap font files — **makebdf**(1), 1
- converting to PCF — **bdftopcf**(1), 1
- converting to SNF — **bdftosnf**(1), 1
- display utility — **xfd**(1), 1
- fonts.dir file — **mkfontdir**(1), 1
- property display utility — **xprop**(1), 1
- selection utility — **xfontsel**(1), 1
- server font listing utility — **xlsfonts**(1), 1
- show contents of SNF file — **showsnf**(1), 1

fs(1) — font server for X Window System, 1

fsadmin(1) — administration utility for font server, 1

fsinfo(1) — font server information utility, 1

fslsfonts — font list displayer, 1

fstobdf(1) — BDF font generator, 1

I

ico2(6) — improved polyhedron animation demo, 1

ico(6) — rotating polyhedron demo program, 1

images
- rasterfile image format — **rasterfile**(4), 1

imake(1) — **make** utility interface, 1

K

kbd_mode(1) — keyboard translation utility, 1

keyboard
- keytable mapping file — **keytable.map**(4), 1
- resetting input — **kbd_mode**(1), 1
- translation utility — **kbd_mode**(1), 1

keytable.map(4) — keytable mapping file, 1

L

listres(1) — lists widget resources, 1

M

mail
- mailbox monitoring utility — **xbiff**(1), 1
- X System mail utility — **xmh**(1)

makebdf(1) — creates bitmap files from scalable fonts, 1

makedepend(1) — create makefile dependencies, 1

makepsres(1) — build PostScript resource database, 1

maze(6) — automated maze demo, 1

mkfontdir(1) — create fonts.dir file, 1

muncher(6) — pattern drawing demo, 1

O

oclock(1) — analog clock, 1

P

plaid(6) — draws plaid-like patterns, 1

PostScript
- command-line interface — **dpsexec**(6), 1
- create C procedures — **pswrap**(1), 1
- encapsulated PostScript display program — **xepsf**(6), 1
- general information about DPS — **dps**(7), 1
- interface — **cps**(1), 1
- rasterfile conversion — **ras2ps**(1), 1
- resource database builder — **makepsres**(1), 1

pswrap(1) — creates C procedures from PostScript code, 1

puzzle(6) — puzzle game demo, 1

R

ras2ps(1) — convert rasterfile to PostScript, 1

rasterfile, 1

redxblue(1) — swaps red & blue in rasterfiles, 1

resize(1) — resets terminal settings to window size, 1

rgb(1) — build colorname database, 1

S

sessreg(1) — manage utmp / wtmp entries, 1
showfont(1) — font dumper, 1
showrgb(1) — display the colorname database, 1
showsnf(1) — show contents of SNF file, 1

T

text
  copy selection to buffer — **xcutsel**(1), 1
texteroids(6) — spinning text game, 1
twm(1) — tab window manager, 1

V

viewres(1) — display widget class hierarchy, 1

W

window
  check system protocols — **winsysck**(1), 1
  dump image of X window — **xwd**(1), 1
  information display utility — **xwininfo**(1), 1
  listing utility — **xlswins**(1), 1
  print X window dump — **xpr**(1), 1
  property display utility — **xprop**(1), 1
  protocol display demo — **xscope**(6), 1
  root window settings — **xsetroot**(1), 1
  set terminal settings to window size — **resize**(1), 1
  tab window manager — **twm**(1), 1
  X Window System — **X11**(7), 1
winsysck(1) — check window protocols, 1
worm(6) — worm drawing demo, 1

X

X11(7) — general infomation about X
xauth(1) — Xserver authorization utility, 1
xbiff(1) — mailbox monitoring utility, 1
xcalc91) — desktop calculator tool, 1
xclipboard(1) — clipboard display utility, 1
xclock(1) — desktop clock, 1
xcmsdb(1) — manipulate color characterization, 1
xcolor(1) — display current colors, 1
xcolsole(1) — display console messages, 1
xcutsel(1) — copy selection to buffer, 1
xditview(1) — display ditroff files, 1
xdm(1) — X display manager, 1
xdpr(1) — X window dump-to-printer utility, 1
xdpyinfo(1) — Xserver information utility, 1
xedit(1) — simple text editor, 1
xepsf(6) — encapsulated PostScript display demo, 1
xev(6) — prints contents of X events, 1
xfd(1) — font display utility, 1
xfontsel(1) — font selection utility, 1
xgc(6) — graphics primitives demo, 1
xhost(1) — controls hosts connected to the Xserver, 1
xinit(1) — Xserver initialization program, 1
xkill(1) — forces Xserver to close connections, 1
xload(1) — display system load average, 1
xlock(1) — screen locking utility, 1
xlogo(1) — display X Sysyem logo, 1
xlsatoms(1) — list interned atoms on server, 1
xlsclients(1) — list client application, 1
xlsfonts(1) — server font list displayer, 1
xlswins(1) — X window listing utility, 1
xmac(6) — MacPaint file display demo, 1
xmag(1) — screen maginification utility, 1
xmag_multivis(6) — screen maginifcation demo, 1
xman(1) — display manpages for X, 1
xmh(1) — X Systems mail utility
xmkmf(1) — interface to **imake** utility, 1
xpr(1) — prints X window dump, 1
xprop(1) — display window & font properties, 1
xrdb(1) — Xserver resource database utility, 1
XReadScreen(3) — returns the displayed colors in a rectangle of the screen., 1
xrefresh(1) — refresh all or part of an X screen, 1
xscope(6) — X window protocol display demo, 1
Xserver
  authorization display utility — **xauth**(1), 1
  client listing application — **xlsclients**(1), 1
  closing connections — **xkill**(1), 1
  display manager — **xdm**(1), 1
  event printing demo — **xev**(6), 1
  general information — **Xserver**(1), 1
  hosts allowed to make connections — **xhost**(1),

1
Xserver, *continued*
information display utility — **xdpyinfo**(1), 1
initialization program — **xinit**(1), 1
listing interned atoms — **xlsatoms**(1), 1
property display utility — **xprop**(1), 1
protocol display program — **xscope**(6), 1
database utility — **xrdb**(1), 1
Solaris implementation — **Xsun**(1), 1
xset(1) — user preference utility, 1
xsetroot(1) — sets parameters of root's wiondow, 1
xsol(6) — solitaire card game, 1
XSolarisOvlCopyAreaAndPaintType(3) — copies the given area and paint type data, 1
XSolarisOvlCopyPaintType(3) — renders opaque and transparent paint, 1
XSolarisOvlCreateWindow(3) — creates an overlay window, 1
XSolarisOvlGetPaintType(3) — gets the current paint type set in the GC., 1
XSolarisOvlIsOverlayWindow(3) — indicates whether a given window is an overlay window, 1
XSolarisOvlSelectPair(3) — selects an optimal overlay/underlay visual pair that best meets the criteria., 1
XSolarisOvlSelectPartner(3) — returns the overlay/underlay visual that best meets the criteria, 1
XSolarisOvlSetPaintType(3) — specifies the type of paint rendered by subsequent Xlib drawing with the given GC., 1
XSolarisOvlSetWindowTransparent(3) — sets the background state of an overlay window to be transparent., 1
xstdcmap(1) — define standard colormap properties, 1
Xsun(1) — Solaris Xserver, 1
xterm(1) — terminal emulator for X, 1
xwd(1) — dump image of X window, 1
xwininfo(1) — X window information utility, 1
xwud(1) — image display utility, 1